

**Towards the Quality of Service for VoIP traffic in IEEE 802.11  
Wireless Networks**

**Sangho Shin**

**Henning Schulzrinne**

**Email: sangho, hgs@cs.columbia.edu**

**Department of Computer Science  
Columbia University**

2008

## ABSTRACT

The usage of voice over IP (VoIP) traffic in IEEE 802.11 wireless networks is expected to increase in the near future due to widely deployed 802.11 wireless networks and VoIP services on fixed lines. However, the quality of service (QoS) of VoIP traffic in wireless networks is still unsatisfactory. In this thesis, I identify several sources for the QoS problems of VoIP traffic in IEEE 802.11 wireless networks and propose solutions for these problems.

The QoS problems discussed can be divided into three categories, namely, user mobility, VoIP capacity, and call admission control. User mobility causes network disruptions during handoffs. In order to reduce the handoff time between Access Points (APs), I propose a new handoff algorithm, Selective Scanning and Caching, which finds available APs by scanning a minimum number of channels and furthermore allows clients to perform handoffs without scanning, by caching AP information. I also describe a new architecture for the client and server side for seamless IP layer handoffs, which are caused when mobile clients change the subnet due to layer 2 handoffs.

I also present two methods to improve VoIP capacity for 802.11 networks, Adaptive Priority Control (APC) and Dynamic Point Coordination Function (DPCF). APC is a new packet scheduling algorithm at the AP and improves the capacity by balancing the uplink and downlink delay of VoIP traffic, and DPCF uses a polling based protocol and minimizes the bandwidth wasted from unnecessary polling, using a dynamic polling list. Additionally, I estimated the capacity for VoIP traffic in IEEE 802.11 wireless networks via theoretical analysis, simulations, and experiments in a wireless test-bed and show how to avoid mistakes in the measurements and comparisons.

Finally, to protect the QoS for existing VoIP calls while maximizing the channel utilization, I propose a novel admission control algorithm called QP-CAT (Queue size Prediction using Computation of Additional Transmission), which accurately predicts the impact of new voice calls by virtually transmitting virtual new VoIP traffic.

# Contents

<b>List of Figures</b>	<b>v</b>
<b>List of Tables</b>	<b>ix</b>
<b>Chapter 1 Introduction</b>	<b>1</b>
1.1 QoS problems for VoIP traffic in wireless networks . . . . .	2
1.2 Original contributions . . . . .	4
1.3 Background . . . . .	6
1.3.1 Architecture of IEEE 802.11 WLANs . . . . .	6
1.3.2 The IEEE 802.11 MAC protocol . . . . .	6
1.3.3 Distributed Coordination Function (DCF) . . . . .	6
1.3.4 Point Coordination Function (PCF) . . . . .	7
1.3.5 IEEE 802.11e MAC enhancements . . . . .	7
1.3.6 IEEE 802.11 standards . . . . .	8
<b>I QoS for User Mobility</b>	<b>10</b>
<b>Chapter 2 Reducing MAC Layer Handoff Delay by Selective Scanning and Caching</b>	<b>11</b>
2.1 Standard layer 2 handoff . . . . .	11
2.1.1 Layer 2 handoff procedure . . . . .	11
2.1.2 Layer 2 handoff time . . . . .	12
2.2 Fast layer 2 handoff algorithm . . . . .	13
2.2.1 Selective Scanning . . . . .	14
2.2.2 Caching . . . . .	14
2.3 Implementation . . . . .	16
2.4 Experiments . . . . .	18
2.4.1 Experimental setup . . . . .	18
2.4.2 Experimental environment . . . . .	18
2.4.3 Measurement . . . . .	18
2.4.4 Experimental results . . . . .	19
2.5 Related work . . . . .	21
2.6 Conclusion . . . . .	23

<b>Chapter 3 Reducing IP Layer Handoff Delay by Fast Subnet Detection and Temporary IP address</b>	<b>25</b>
3.1 Introduction . . . . .	25
3.2 Layer 3 handoff algorithm . . . . .	25
3.2.1 Fast subnet change detection . . . . .	26
3.2.2 Discovering the temporary IP address . . . . .	27
3.2.3 Session updates . . . . .	29
3.3 Implementation . . . . .	30
3.4 Experiments . . . . .	32
3.4.1 Test environment . . . . .	32
3.4.2 Parameter calculation . . . . .	32
3.4.3 Measurements . . . . .	32
3.4.4 Experimental results . . . . .	33
3.5 Related work . . . . .	35
3.6 Conclusion . . . . .	36
<b>Chapter 4 Passive Duplicate Address Detection for DHCP</b>	<b>38</b>
4.1 Introduction . . . . .	38
4.2 Standard DHCP procedure . . . . .	39
4.3 Framework of pDAD . . . . .	40
4.3.1 AUC . . . . .	40
4.3.2 DHCP server behavior . . . . .	41
4.4 Experiments . . . . .	41
4.4.1 Experimental setup . . . . .	42
4.4.2 Experimental results . . . . .	42
4.5 Conclusion . . . . .	45
<b>II QoS and VoIP Capacity</b>	<b>49</b>
<b>Chapter 5 The VoIP Capacity of IEEE 802.11 WLANs</b>	<b>50</b>
5.1 Introduction . . . . .	50
5.2 Theoretical capacity for VoIP traffic . . . . .	50
5.2.1 Capacity for CBR VoIP traffic . . . . .	50
5.2.2 Capacity for VBR VoIP traffic . . . . .	52
5.3 Capacity for VoIP traffic via simulation . . . . .	54
5.3.1 Simulation parameters . . . . .	54
5.3.2 Capacity for CBR VoIP traffic . . . . .	55
5.3.3 Capacity for VBR VoIP traffic . . . . .	55
5.4 Capacity for VoIP traffic via experiments . . . . .	55
5.4.1 The ORBIT test-bed . . . . .	56
5.4.2 Experimental results . . . . .	57
5.4.3 Analysis of the results and comparison with simulation results . . . . .	57
5.5 Factors that affect the experimental and simulation results . . . . .	59
5.5.1 Preamble size . . . . .	59
5.5.2 Rate control . . . . .	60

5.5.3	VoIP packet generation offsets among VoIP sources . . . . .	61
5.5.4	Channel transmission rate of Acknowledgment (ACK) frames . . . . .	63
5.5.5	Received Signal Strength Indication (RSSI) . . . . .	63
5.5.6	Scanning APs . . . . .	65
5.5.7	Retry limit . . . . .	66
5.5.8	Network buffer size . . . . .	66
5.6	Experimental capacity for VoIP traffic with 802.11e . . . . .	68
5.6.1	Capacity for VoIP traffic in IEEE 802.11e . . . . .	68
5.6.2	Effect of TCP traffic on VoIP traffic . . . . .	69
5.6.3	Impact of each 802.11e parameter . . . . .	72
5.7	Related work . . . . .	73
5.8	Conclusion . . . . .	76
<b>Chapter 6</b>	<b>Improving VoIP Capacity in PCF: Dynamic PCF</b>	<b>77</b>
6.1	Introduction . . . . .	77
6.2	Problems of using PCF for VBR VoIP traffic . . . . .	77
6.2.1	Polling during silence period . . . . .	77
6.2.2	Synchronization between polls and data . . . . .	78
6.2.3	Multiple packetization intervals . . . . .	78
6.3	Dynamic Point Coordination Function (DPCF) . . . . .	78
6.3.1	DPCF at the AP: Dynamic Polling List . . . . .	78
6.3.2	DPCF at client side . . . . .	81
6.3.3	Differentiation of traffic types . . . . .	83
6.4	Simulations . . . . .	83
6.5	Results and analysis . . . . .	83
6.5.1	Capacity of VoIP traffic . . . . .	83
6.5.2	Capacity of VoIP with data traffic . . . . .	83
6.6	Comparison with IEEE 802.11e . . . . .	85
6.7	Related work . . . . .	87
6.8	Conclusions . . . . .	88
<b>Chapter 7</b>	<b>Improving VoIP Capacity in DCF: Adaptive Priority Control</b>	<b>89</b>
7.1	Introduction . . . . .	89
7.2	Adaptive Priority Control (APC) . . . . .	89
7.2.1	Priority mechanisms at the MAC layer . . . . .	90
7.2.2	APC algorithm . . . . .	91
7.3	Theoretical analysis . . . . .	92
7.3.1	Non-empty client queues ( $Q_C \geq 1$ ) . . . . .	93
7.3.2	Empty client queues $Q_C = 0$ . . . . .	93
7.4	Simulation results . . . . .	95
7.4.1	Evaluation of APC . . . . .	95
7.4.2	Comparison with CW control method . . . . .	99
7.5	Implementation and experiments . . . . .	101
7.5.1	Implementation . . . . .	101
7.5.2	Experiments . . . . .	101

7.6	APC and IEEE 802.11e . . . . .	103
7.7	APC without knowledge of the client queue size . . . . .	105
7.7.1	Estimating the queue size of the client . . . . .	105
7.7.2	Controlling the downlink delay with PID control . . . . .	106
7.8	Related work . . . . .	109
7.9	Conclusion . . . . .	109

### III QoS and Call Admission Control 111

#### Chapter 8 Call Admission Control using Queue size Prediction by Computing Additional Transmissions (QP-CAT) 112

8.1	Introduction . . . . .	112
8.2	Correlation between the queue size and downlink delay . . . . .	112
8.3	Queue size Prediction (QP) using Computation of Additional Transmission (CAT) . . . . .	115
8.3.1	Emulation of new VoIP flows . . . . .	116
8.3.2	Computation of Additional Transmission (CAT) . . . . .	116
8.4	Simulations and their results . . . . .	119
8.4.1	Simulation setup . . . . .	119
8.4.2	Simulation results . . . . .	120
8.5	Implementation and experiments . . . . .	122
8.5.1	Implementation . . . . .	123
8.5.2	Experimental setup . . . . .	124
8.5.3	Experimental results . . . . .	124
8.6	Extensions for QP-CAT . . . . .	125
8.6.1	QP-CAT with IEEE 802.11e (QP-CATe) . . . . .	125
8.6.2	Running multiple instances of QP-CAT . . . . .	126
8.7	Related work . . . . .	127
8.7.1	Comparison with other CAC methods . . . . .	128
8.8	Conclusion . . . . .	129

#### Chapter 9 Conclusion 130

9.1	QoS for user mobility . . . . .	130
9.2	VoIP capacity . . . . .	131
9.3	Call admission control . . . . .	132

# List of Figures

1.1	VoIP traffic over IEEE 802.11 wireless networks . . . . .	2
1.2	Problem domain of the VoIP traffic in IEEE 802.11 WLANs . . . . .	3
1.3	Architecture of IEEE 802.11 WLANs . . . . .	5
1.4	DCF MAC behavior . . . . .	6
1.5	PCF . . . . .	7
1.6	IEEE 802.11e HCF . . . . .	8
2.1	Layer 2 handoff process in IEEE 802.11 . . . . .	12
2.2	Layer 2 handoff time in IEEE 802.11b . . . . .	13
2.3	Channels used in IEEE 802.11b . . . . .	13
2.4	Selective scanning procedure . . . . .	15
2.5	Caching procedure . . . . .	17
2.6	Layer 2 handoff time in 802.11b . . . . .	19
2.7	Packet loss at the mobile receiver and packet delay at the mobile sender . . . . .	22
2.8	Handoff time in IEEE 802.11a . . . . .	23
3.1	Fast layer 3 handoff procedure . . . . .	26
3.2	Enhanced cache structure . . . . .	26
3.3	Average IP usage in the Columbia University wireless network . . . . .	28
3.4	SIP session update . . . . .	29
3.5	Full SIP session update . . . . .	30
3.6	Full layer 3 handoff under scenario one (no lease) . . . . .	31
3.7	The architecture of the fast L3 handoff implementation . . . . .	31
3.8	Experimental environment . . . . .	32
3.9	L3 handoff time using new approach . . . . .	34
3.10	Messages exchanged during L3 handoff and delay components . . . . .	35
4.1	Outline of Passive DAD operation . . . . .	38
4.2	DHCP procedure . . . . .	39
4.3	Framework of pDAD . . . . .	40
4.4	Structure of entries in the AUC's table . . . . .	41
4.5	Structure of packets sent by the AUC to the DHCP server . . . . .	41
4.6	Experimental setup . . . . .	42
4.7	Number of new IP addresses detected by DHCP . . . . .	43
4.8	Traffic volume between DHCP server and relay agent . . . . .	44

4.9	Cumulative distribution function of number of packets per second DHCP server received . . . . .	45
4.10	ARP and broadcast traffic volume to the AUC . . . . .	46
4.11	CDF of the ARP and broadcast traffic to the AUC . . . . .	46
4.12	Timeline of CPU load of AUC and traffic volume received by AUC . . . . .	47
4.13	Cumulative distribution function of CPU load of AUC . . . . .	47
5.1	Conversational speech model in ITU-T P.59 . . . . .	53
5.2	Simulation topology . . . . .	54
5.3	90th percentile delay and retry rate of CBR VoIP traffic in simulations . . . . .	55
5.4	90th percentile delay and retry rate of VBR VoIP traffic in simulations . . . . .	56
5.5	Node layout in the grid ORBIT test-bed . . . . .	57
5.6	90th percentile delay and retry rate of CBR VoIP traffic in the experiments . . . . .	58
5.7	90th percentile delay and retry rate of VBR VoIP traffic in the experiments . . . . .	58
5.8	90th percentile delay and retry rate of CBR VoIP traffic with long and short preamble via experiments . . . . .	60
5.9	90th percentile delay and retry rate of CBR VoIP traffic with and without the AMRR rate control algorithm in the experiments . . . . .	61
5.10	An example of VoIP packet transmission in the application layer with 10 VoIP sources with the fixed transmission offset of $x$ . . . . .	62
5.11	90th percentile delay and retry rate as a function of packet generation offset among VoIP sources . . . . .	62
5.12	Experimental results with different ACK transmission rates . . . . .	64
5.13	RSSI values as a function of the distances between nodes and the AP . . . . .	65
5.14	Probe request frames and retry rate in the experiments . . . . .	66
5.15	Distribution and cumulative distribution function (CDF) of number of retransmissions of CBR VoIP traffic in the experiments . . . . .	67
5.16	90th percentile of delay of CBR VoIP traffic in AC_VO and AC_VI modes of 802.11e . . . . .	68
5.17	Retry rate of CBR VoIP traffic in AC_VO and AC_VI in 802.11e . . . . .	69
5.18	90th percentile of delay of VBR VoIP traffic in each 802.11e AC . . . . .	70
5.19	Retry rate of VBR VoIP traffic in each 802.11e AC . . . . .	70
5.20	Experimental results of CBR VoIP traffic (AC_VO) with 1 to 3 TCP sources (AC_BK): 90th percentile of delay of the VoIP traffic and the total throughput of TCP traffic . . . . .	71
5.21	Experimental results of CBR VoIP traffic (AC_VO) with 1 to 3 TCP sources (AC_BE): 90th percentile of delay of the VoIP traffic and the total throughput of TCP traffic . . . . .	71
5.22	Experimental results of VBR VoIP traffic (AC_VO) with 1 to 3 TCP sources (AC_BK): 90th percentile of delay of the VoIP traffic and the total throughput of TCP traffic . . . . .	72
5.23	The effect of each 802.11e parameter; delay is 90th percentile (refer to Table 5.3 for the experimental parameters in each case.) . . . . .	74
5.24	The effect of each 802.11e parameter on total throughput and packet loss rate . . . . .	75
6.1	Synchronization problem in PCF . . . . .	79
6.2	DPCF algorithm at the AP side . . . . .	80
6.3	DPCF algorithm at the client side . . . . .	82
6.4	Packet transfer in DPCF mode with 20 ms CFP interval . . . . .	82
6.5	90th percentile of end-to-end delay of VoIP in each MAC protocol . . . . .	84



6.6	Number of polls and Null function frames in PCF and DPCF with 30 VBR VoIP clients . .	85
6.7	90th percentile of delay of VoIP traffic and throughput of TCP traffic with 28 VoIP clients and 1 to 3 TCP flows . . . . .	86
6.8	Simulation results of EDCA with TCP traffic in NS-2 . . . . .	87
7.1	The uplink and downlink delay of VoIP traffic in DCF . . . . .	90
7.2	Packet transmission in APC . . . . .	92
7.3	Optimal P value when $Q_C = 0$ . . . . .	94
7.4	Delay as a function of the number of VoIP sources using two priority control methods with 20 ms packetization interval 64 kb/s VoIP traffic . . . . .	96
7.5	Delay as a function of the number of VoIP sources using two priority control methods with mixed (10 ms and 20 ms) packetization interval 64 kb/s VBR VoIP traffic . . . . .	97
7.6	Delay as a function of the number of VoIP sources using VoIP traffic with 40 ms packeti- zation intervals . . . . .	98
7.7	Delay as a function of the number of VoIP sources using VoIP traffic with 20 ms and 40 ms packetization intervals . . . . .	98
7.8	The uplink and downlink delay as a function of simulation time with 36 VBR VoIP sources using APC . . . . .	99
7.9	Delay as a function of the number of VoIP sources using CW to control the transmission rate	100
7.10	Retry rate of the AP (downlink traffic) as a function of the number of VoIP sources in three approaches: Controlling CW, DCF, and APC . . . . .	100
7.11	Delay as a function of the number of CBR VoIP sources in the experiments . . . . .	102
7.12	CFT values at the driver and actual number of packets sent contention free (the frequency and CDF) . . . . .	102
7.13	Delay and throughput as a function of the number of VoIP sources with TCP traffic in the experiments . . . . .	103
7.14	Delay and retry rate as a function of the number of VoIP sources in IEEE 802.11e . . . .	104
7.15	Delay as a function of the number of VoIP sources using the estimated queue size of clients	106
7.16	Effect of P and D terms on delay . . . . .	107
7.17	PID controller in APC . . . . .	107
7.18	Delay as a function of the number of VoIP sources of APC with the control method . . . .	108
8.1	Correlation between the queue size of the AP and instant downlink delay in different num- ber of VoIP sources: each point represents the downlink delay of a frame and the queue size of the AP when the frame was queued, and the straight line is the theoretical model. .	113
8.2	The errors between the actual downlink delay and the estimated delay using the queue size of the AP . . . . .	114
8.3	Basic concept of QP-CAT . . . . .	115
8.4	Emulation of a new VoIP flow with 20 ms packetization interval . . . . .	116
8.5	Computing the number of additionally transmittable VoIP packets . . . . .	116
8.6	Handling the remaining time ( $T_r$ ): when $T_r > T_b$ . . . . .	117
8.7	Handling the remaining time ( $T_r$ ): when $T_r \leq T_b$ . . . . .	118
8.8	Emulation of collisions: during $2T_t$ , only one additional frame can be transmitted due to the collision, in the end . . . . .	119
8.9	Flowchart of QP-CAT algorithm . . . . .	120

8.10	Simulation results of QP-CAT with 32 kb/s VoIP traffic using 20 ms packetization interval .	121
8.11	Simulation results of QP-CAT with various types of CBR VoIP traffic . . . . .	121
8.12	Simulation results of QP-CAT with 14 CBR VoIP sources with increasing the collision rate	122
8.13	Simulation results of QP-CAT with 64 kb/s VBR VoIP traffic using 20 ms packetization interval . . . . .	123
8.14	Experimental results of QP-CAT with 64 kb/s and 20 ms packetization interval CBR VoIP traffic in various channel status . . . . .	124
8.15	Experimental results of QP-CAT for 32 kb/s and 40 ms packetization interval CBR VoIP traffic; the capacity for the VoIP traffic was 28 calls . . . . .	125
8.16	QP-CATe: when background traffic is transmitted before using up the TXOP of the AP, the remaining TXOP duration is considered as $T_C$ . . . . .	125
8.17	Experimental results of QP-CATe with 64 kb/s and 20 ms packetization interval CBR VoIP traffic and a TCP flow; the capacity for the VoIP traffic was 15 calls . . . . .	126

# List of Tables

1.1	Parameters of IEEE 802.11e . . . . .	8
2.1	Cache structure . . . . .	16
2.2	Handoff delay (ms) in the experiments . . . . .	19
2.3	Handoff time in the environment without rogue APs (ms) . . . . .	20
2.4	Packet delay (ms) during handoff in mobile sender . . . . .	20
2.5	The number of packets lost during handoff in mobile receiver . . . . .	21
2.6	Bridging delay . . . . .	21
3.1	IP address acquisition time in normal DHCP and the new approach . . . . .	33
3.2	Packet loss during L3 handoff using the new approach . . . . .	35
4.1	Observed umber of MAC addresses with multiple IP addresses . . . . .	43
5.1	Parameters in IEEE 802.11b (11 Mb/s) . . . . .	52
5.2	Voice pattern parameters in ITU-T P.59 . . . . .	53
5.3	Experimental configuration . . . . .	73
7.1	Packet transmission in APC with 802.11e . . . . .	105
8.1	IEEE 802.11b parameters (11 Mb/s) . . . . .	117
8.2	Comparison of CAC methods . . . . .	128
9.1	Total handoff time using combined solutions . . . . .	130

# Chapter 1

## Introduction

As many IEEE 802.11 wireless networks have been widely deployed, the importance of VoIP over the wireless networks has been increasing, motivating efforts to improve Quality of Service (QoS) for VoIP traffic.

Since the first standard for IEEE 802.11 Wireless Local Area Networks (WLANs) was introduced in 1999, 802.11 WLANs have been gaining in popularity. Most of the mobile devices such as laptops and PDAs support 802.11, and WLANs also have been deployed in places like coffee shops, air ports, and shopping malls.

The main reasons for its popularity are as follows: First, IEEE 802.11 WLAN uses unlicensed channels in the 2.4 GHz and 5.0 GHz bands. Second, the deployment is very easy and its cost is also low. Lastly, it supports high speed data transmission; 802.11b supports 11 Mb/s, and 802.11a/g support 54 Mb/s data transmission. The recent IEEE 802.11n draft supports more than 100 Mb/s using Multi-Input Multi-Output (MIMO) technology, and very recently 802.11VHT (Very High Throughput) study group was formed in IEEE targeting a throughput of 1 Gb/s. For the above reasons, recently, many cities have been deploying freely accessible Access Points (APs) in streets and parks so that people can use wireless networks free without any subscription to the service, which allows people to connect to the Internet anywhere and anytime.

Due to the fast growth of IEEE 802.11-based wireless LANs during the last few years, Voice over IP (VoIP) became one of the most promising services to be used in mobile devices over WLANs. VoIP has been replacing the traditional phone system because of easy development, reduced cost, and advanced new services, and the successful deployment of VoIP service in fixed lines is being expanded to VoIP over WLANs.

VoIP in IEEE 802.11 WLANs means that users send voice data through IEEE 802.11 WLAN technology to the AP. As we can see in Fig. 1.1, the mobile VoIP client associates with an AP, and the AP is connected to the Internet in different ways. Users can call other mobile clients, fixed IP phones, and even traditional phones connected via IP gateways. Many companies produce VoIP wireless phones or PDAs that support both the cellular and 802.11 wireless networks, and very recently a major cellular phone service provider started a service plan that allows users to call through both cellular and WLANs. Therefore, the number of wireless VoIP users is expected to increase in the near future.

However, in spite of the expected increase of wireless VoIP users, the Quality of Service (QoS) of VoIP traffic in WLANs does not meet the growth. According to ITU-T Recommendation G.114 [29], one way transmission delay for a good quality of service needs to be less than 150 ms, and in WLANs the

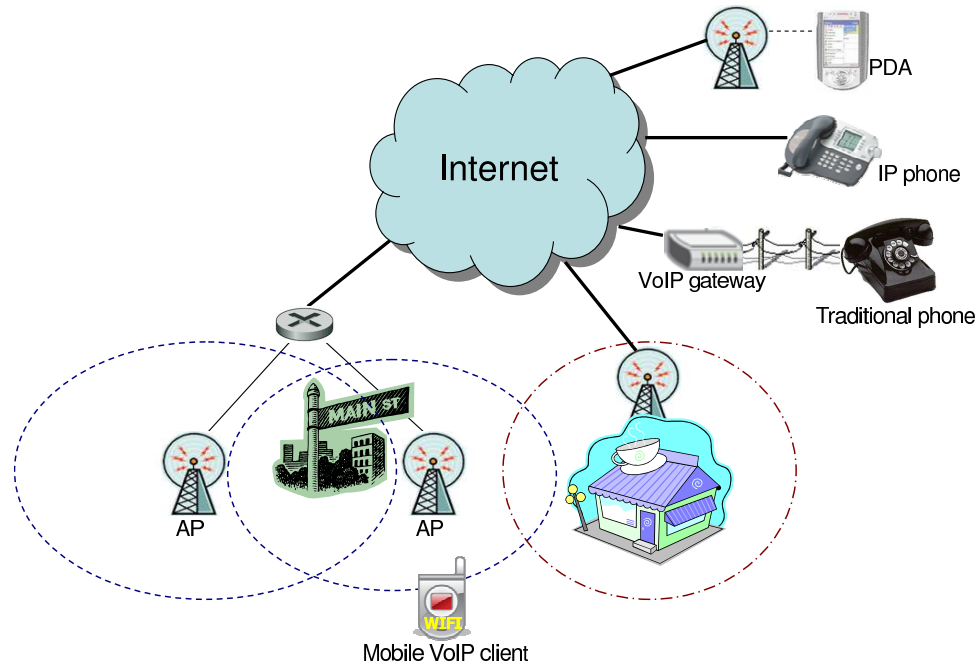


Figure 1.1: VoIP traffic over IEEE 802.11 wireless networks

one way delay between the AP and clients for the good voice quality needs to be lower than 60 ms [35], considering that the network delay is 30 ms and the encoding and decoding delay at VoIP clients is 30 ms each. However, the delay easily exceeds the limit for various reasons in WLANs, as explained in the next section. Even though the IEEE 802.11e [26] standard has been introduced in 2005 to support better quality of service for real times services like voice and video, it just gives higher priority to such traffic against background traffic and still does not solve many QoS issues. In the next sections, I identify QoS issues on VoIP in IEEE 802.11 WLANs and explain my contributions towards solutions to these problems.

## 1.1 QoS problems for VoIP traffic in wireless networks

The QoS problems can be divided largely into three large categories, namely, handoff, capacity, and call admission control (Fig. 1.2). Some of the problems differ from those in fixed line VoIP service, and some are the same, but the solutions are totally different. The handoff problems are new ones and do not occur in wired networks, the capacity and call admission control issues are shared, but the approaches for solutions differ from those in wired VoIP service.

The first problem, handoff, is caused by the mobility of users. As shown in Fig. 1.1, wireless clients associate with each AP and exchange VoIP packets via the AP. The coverage range of an AP is limited, and wireless clients need to change the AP when they move out of the coverage of the AP they are currently associated with. The procedure of moving to a new AP is called handoff, and during the handoff, the network connectivity is disrupted and voice communication is broken. The handoff is divided into two types, layer 2 handoff and layer 3 handoff. Layer 2 handoff is also called MAC layer handoff and happens when wireless clients move between two APs within the same subnet. If the subnet changes due to layer

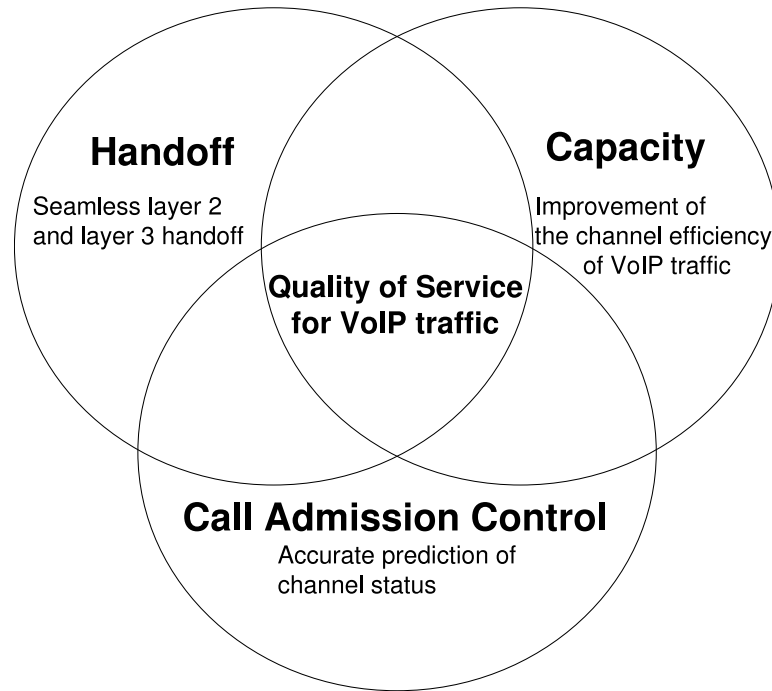


Figure 1.2: Problem domain of the VoIP traffic in IEEE 802.11 WLANs

2 handoffs, a layer 3 handoff also needs to be performed. When the subnet changes, the old IP address becomes invalid, and clients need to acquire new IP addresses in the new subnet. Thus, the layer 3 handoff is also called IP layer handoff. Acquiring an IP address involves interaction between the Dynamic Host Configuration Protocol (DHCP) [12] server and clients, which makes the handoff time longer than that in layer 2 handoff. Additionally, when IP addresses change, all sessions in the clients need to be updated with the new IP address, unless Mobile IP is used. The session update needs to be handled by each application, and it is called application layer handoff. However, I include the session update in the layer 3 handoff because the IP address change is meaningless without the session update. For these reasons, even though layer 3 handoff does not happen frequently, it takes a long time or is not supported in some operating systems and devices, and thus it is critical for real time services.

The second problem, the capacity issue, is caused by the need to support a large number of concurrent voice conversations in public spaces such as airports, train stations, and stadiums, and by the constraint that a limited number of channels and APs can be installed in a certain space due to the limited number of non-overlapping channels in 802.11 WLANs. The capacity for VoIP traffic in WLANs is much lower than that in Ethernet. The first reason is that the bandwidth of WLAN is lower than that of Ethernet. Even though WLANs support up to 54 Mb/s with the introduction of 802.11g [24], it is still much lower than that of a typical fixed line. IEEE 802.11n [27] supports 100 Mb/s using Multi-Input-Multi-Output (MIMO), but it would be difficult to achieve such a speed in a crowded city, where all channels are fully occupied with other APs and clients, because it needs to use multiple channels simultaneously. Another reason for the low capacity is that the total throughput of VoIP traffic is far below the nominal bit rate due to the overhead of VoIP packet transmission in WLANs. If we look at a VoIP packet in WLANs, the voice payload takes

only 18% <sup>1</sup> and the remaining 82% of a VoIP packet is the overhead to transmit the packet. Considering that more than half of the overhead is caused in the MAC layer, we need to improve the voice capacity by eliminating the overhead at the MAC layer (Section 6).

The final problem is call admission control. When the number of flows in a Basic Service Set (BSS) exceeds the capacity of the channel, the overall QoS of all flows drastically deteriorates. Thus, when the number of current calls reaches the capacity, further calls need to be blocked or forwarded to another channel or AP using call admission control. The admission control in WLANs totally differs from that in wired networks because the bottleneck is not the router capacity, but the wireless channel capacity between the AP and clients in WLANs. Therefore, the biggest challenge for the call admission control in WLANs is to identify the impact of new VoIP flows on the channel. It is very difficult to predict it because the channel capacity changes according to various factors, such as the data rate of clients, RF interference, and retransmission rate. If the instant channel capacity is overestimated, too many voice calls are admitted and the QoS of existing calls is deteriorated, and if it is underestimated, bandwidth is wasted and the overall voice capacity decreases. Therefore, the ultimate goal for call admission control in WLANs is to protect the QoS of existing voice calls, minimizing the wasted bandwidth.

## 1.2 Original contributions

In this section, I explain my contributions that I have achieved through my study on the QoS of VoIP traffic in WLANs. First, I have achieved a seamless layer 2 handoff using Selective Scanning and Caching (Section 2). Usually, layer 2 handoff takes up to 500 ms because it takes a long time to scan all channels to find new APs. I have reduced the scanning time to 100 ms using Selective Scanning, where clients scan the channels on which new APs are likely installed. Furthermore, I reduced the handoff time to a few milliseconds using Caching, where clients store the scanned AP information in a cache, and they can perform handoffs without scanning using the cached information. Many solutions have been proposed in the past, but most of them require changing APs or infrastructure, or they need to change the standard, which requires a modification of the firmware, and thus they are not practically deployable. However, my solutions requires only changes on the client side, specifically, wireless card drivers of clients.

Second, I have improved the total layer 3 handoff including session update to 40 ms, and 200 ms in the worst case. Generally, layer 3 handoff takes up to a few seconds because there is no standard way to detect the subnet change, and also because it takes up to 1 second to acquire a new IP address in the new subnet. I have introduced a fast subnet change detection method, which takes only one packet round trip time (20 ms in experiments). Also, in order to avoid the network disruption due to the long IP address acquisition time, which is one second using the standard DHCP, I proposed a TEMP\_IP approach, which reduces the network disruption to only 130 ms. Mobile IP [59][60] has been proposed and a lot of research have been done to improve the performance for the last ten years, but still it is not deployed in many places yet for practical reasons. As in the seamless layer 2 handoff algorithm, the proposed layer 3 handoff algorithm requires only a change in the client. Also, I have reduced the new IP address assignment time of DHCP server, by using Passive Duplicate Address Detection (pDAD). When the DHCP server assign a new IP address to a client, it checks if the IP address is used by other clients or not, by sending an ICMP echo, and it waits the response for up to a second. In pDAD, the DHCP server monitors all the usage of IP addresses in the subnet in real time, so that it can assign new IP addresses to clients promptly without additional duplicate address detection. pDAD can also detect unauthorized use of IP addresses in real time

---

<sup>1</sup>when using 64 kb/s voice traffic with 20 ms packetization interval in DCF

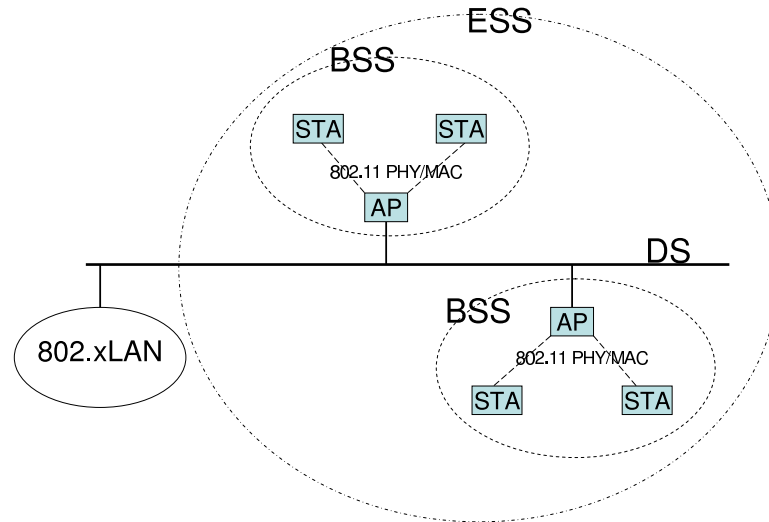


Figure 1.3: Architecture of IEEE 802.11 WLANs

and helps identifying malicious users.

Third, I have measured the VoIP capacity in 802.11 WLANs via experiments and compare it with the capacity measured via simulations and theoretical analysis. I also have identified the factors that have been commonly overlooked but affect the capacity, in experiments and simulations. I also experimentally measured the VoIP capacity using 802.11e and identified how well 802.11e can protect the QoS for VoIP traffic against background traffic. This study can be applied to analyze any 802.11 experimental results, not only for VoIP capacity measurement.

Fourth, I have improved the VoIP capacity using two variation of media protocols, Dynamic Point Coordination Function (DPCF) and Adaptive Priority Control (APC) in the Distributed Coordination Function (DCF), by 25% to 30%. DPCF minimizes the bandwidth wasted by unnecessary polling, which is a big overhead of PCF protocol, by managing the dynamic polling list, which contains active (talking) nodes only. APC balances the uplink and downlink delay by distributing channel resources between uplink and downlink, dynamically adapting to change of the number of VoIP sources and the traffic volumes of uplink and downlink.

Fifth, the QoS of existing calls can be protected more efficiently, maximizing the utilization of channels, using a novel call admission control, QP-CAT. The existing call admission control methods cannot adapt to the change of channel status, and they just reserve some amount of bandwidth for such cases such that the bandwidth is usually wasted. QP-CAT uses the queue size of the AP as the metric, and it predicts the increase in the queue size caused by new VoIP flows accurately, by monitoring the current channel in real time. Also, it can predict the impact of new calls when the background traffic exist together with VoIP traffic under 802.11e.



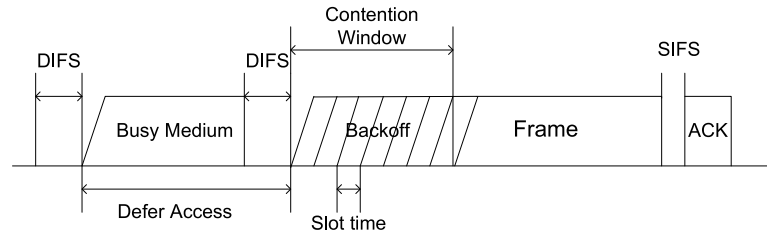


Figure 1.4: DCF MAC behavior

## 1.3 Background

### 1.3.1 Architecture of IEEE 802.11 WLANs

IEEE 802.11 WLAN is defined as local wireless communication using unlicensed channels in the 2.4 GHz and 5 GHz bands. The 802.11 architecture is comprised of several components and services [23].

**Wireless LAN station:** The station (STA) is the most basic component of the wireless network. A station is any device that contains the functionality of the 802.11 protocol: medium access control (MAC), physical layer (PHY), and a connection to the wireless media. Typically, the 802.11 functions are implemented in the hardware and software of a network interface card (NIC). A station could be a laptop PC, handheld device, or an Access Point (AP). All stations support the 802.11 station services of authentication, de-authentication, privacy, and data delivery.

**Basic Service Set (BSS):** The Basic Service Set (BSS) is the basic building block of an 802.11 wireless LAN. The BSS consists of a group of any number of stations.

**Distribution System (DS):** Multiple BSS can form an extended network component, and the distribution system (DS) is used to interconnect the BSSs. Generally Ethernet is used as DS.

**Extended Service Set (ESS):** Using multiple BSS and DS, any size of wireless networks can be created, and such type of network is called Extended Service Set network. Each ESS is recognized using an ESS identification (ESSID), and it is different from subnet. However, in many cases, an ESS comprises a subnet.

### 1.3.2 The IEEE 802.11 MAC protocol

This section gives an overview of the IEEE 802.11 MAC protocols and the IEEE 802.11e enhancements. The IEEE 802.11 standard provides two different channel access mechanisms, DCF and PCF.

### 1.3.3 Distributed Coordination Function (DCF)

DCF (Distributed Coordination Function) is based on the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) channel access mechanism. DCF supports two different transmission schemes. The default scheme is a two-way handshaking mechanism where the destination transmits a positive acknowledgment (ACK) upon successful reception of a packet from the sending STA. This ACK is needed because the STA cannot determine if the transmission was successful just by listening to its own transmission. The second scheme is a four-way handshake mechanism where the sender, before sending any packet reserves the medium by sending a Request To Send (RTS) frame and waits for a Clear To Send (CTS) from the AP in response to the RTS. Only upon receiving the CTS, will the STA start its transmission.

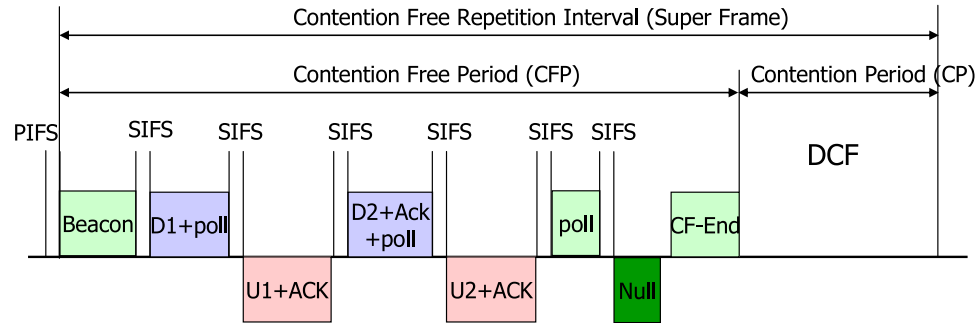


Figure 1.5: PCF

In both schemes, in order to avoid collisions, a backoff mechanism is used by each STA (Fig. 1.4). The STA senses the medium for a constant time interval, the Distributed Interframe Space (DIFS). If the medium is idle for a duration of time equal to DIFS, the STA decreases its own backoff timer. The STA whose backoff timer arrives to zero first transmits. DIFS is used when the frame to be transmitted is a data frame. If the frame to be transmitted is an ACK or a fragment of a previous packet, then the Short Interframe Space (SIFS) is used instead. While the DCF is the fundamental access method used in IEEE 802.11 networks, it does not support Quality of Service (QoS), making this scheme inappropriate for VoIP applications with their stringent delay constraints.

### 1.3.4 Point Coordination Function (PCF)

PCF (Point Coordination Function) is based on a polling mechanism as shown in Fig. 1.5. Each STA is included in a polling list. The Point Coordinator (PC), which is generally the AP, sends a CF-Poll frame to each pollable STA in the polling list. The STA responds by sending a Data frame if it has data to send or a Null function if it has no data to send at that time.

Usually, in an infrastructure network, the AP acts as the PC. When a PC is operating, the two access methods alternate, with a Contention Free Period (CFP) followed by a Contention Period (CP). The PCF is used for frame transfers during a CFP, while the DCF is used for frame transfers during a CP. The PC needs to sense the medium idle for an amount of time equal to Point Interframe Space (PIFS) before gaining access to the medium at the start of the CFP, where  $SIFS < PIFS < DIFS$ .

Piggybacking is commonly used. If the PC has some data to send to a particular pollable STA, a Data + CF-Poll frame will be sent to this STA and the STA will respond with a Data + CF-Ack frame if it has data to send or with CF-Ack (no data) if it does not have any data to send at that time.

### 1.3.5 IEEE 802.11e MAC enhancements

To support applications with Quality of Service (QoS) requirements on IEEE 802.11 networks, the IEEE 802.11e standard has been standardized in 2005 [26]. It introduces the concept of the Hybrid Coordination Function (HCF) for the MAC mechanism. HCF is backward compatible with DCF and PCF, and it provides QoS STAs with prioritized and parameterized QoS access to the wireless medium. The HCF uses both a contention-based channel access method, called the Enhanced Distributed Channel Access (EDCA) and a contention-free channel access method, called HCF Controlled Channel Access (HCCA). With the EDCA, QoS is supported by using four access categories (ACs), each one corresponding to an individual prioritized

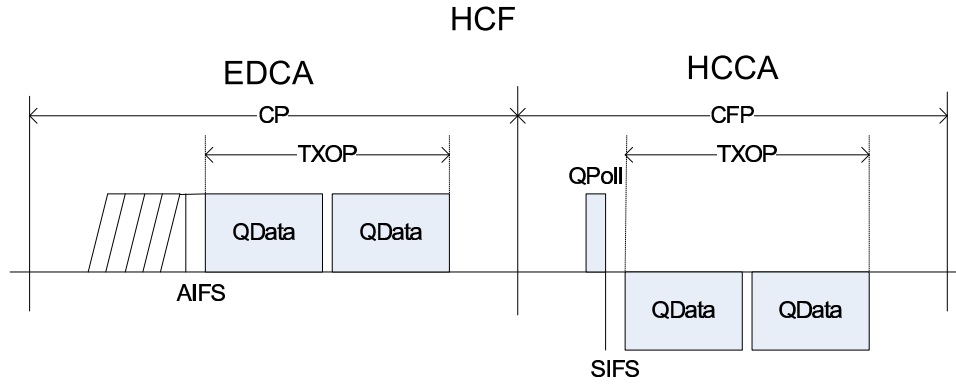


Figure 1.6: IEEE 802.11e HCF

Table 1.1: Parameters of IEEE 802.11e

AC	minimum CW	maximum CW	AIFSN	TXOP ( $\mu$ s)
AC_BK	$aCW_{min}$	$aCW_{max}$	7	0
AC_BE	$aCW_{min}$	$aCW_{max}$	3	0
AC_VI	$(aCW_{min} + 1)/2 - 1$	$aCW_{max}$	2	6016
AC_VO	$(aCW_{min} + 1)/4 - 1$	$(aCW_{max} + 1)/2 - 1$	2	3264

output queue in the STA. A traffic class which requires lower transmission delay can use an AC with higher priority in its contention for the channel. With the HCCA, a hybrid coordinator (HC) allocates transmission opportunities (TXOPs) to wireless STAs by polling, to allow contention-free transfers of data, based on QoS policies. An HC can generate an alternation of contention-free and contention period (Fig. 1.6).

While EDCA is implemented in many commercial wireless cards including the Atheros chipset, the possibility that HCCA will be implemented in commercial wireless cards appears low, as in the case of PCF. Thus, only EDCA is considered in this thesis. EDCA has four access categories (0 to 3) to differentiate traffic. AC0 (AC\_BK) is for background traffic, AC1 (AC\_BE) is for best effort, AC2 (AC\_VI) is for video, and AC3 (AC\_VO) is for voice traffic. Assignment of access category to each traffic is implementation dependent, but generally DSCP (Differentiated Services Codepoint) field [54] in the IP header is used. Traffic is prioritized by different contention window (CW) size, arbitrary interframe spacing (AIFS), and transmission opportunity (TXOP). AIFS is determined by the arbitrary interframe spacing number (AIFSN) as follows:  $AIFS = AIFSN \times aSlotTime + aSIFSTime$ , where AIFSN is defined in the 802.11e standard (Table 1.1), and  $aSlotTime$  and  $aSIFSTime$  are defined in the 802.11a/b/g standards. TXOP is a duration when wireless nodes can transmit frames without backoff; when a wireless node acquires a chance to transmit a frame successfully, it can transmit next frames after only SIFS during the period of TXOP. The parameters for each access category are listed in Table 1.1.  $aCW_{min}$  and  $aCW_{max}$  values are clearly defined in the standard, but generally 31 and 1023 are used, respectively.

### 1.3.6 IEEE 802.11 standards

Since the first IEEE 802.11 standard was introduced in 1999, many 802.11 standards have been released to improve the performance of IEEE 802.11 WLANs. Below, some important standards are explained briefly.

- 802.11a/b/g: Amendment of IEEE 802.11 standard to support high speed networking. 802.11a uses

the 5.4 GHz band and 11 to 13 non-overlapping channels, supporting data rate of up to 54 Mb/s. 802.11b uses 2.4 GHz band and 3 non-overlapping channels among 11 channels, supporting 11 Mb/s. Even though 802.11a was standardized first, 802.11b was commercialized earlier than 802.11a due to technical difficulties. After 802.11a was out in the market, it was not deployed widely because of 802.11g products, which also support 54 Mb/s and are compatible with the widely deployed 802.11b, using the same 2.4 GHz band.

- 802.11e: It was standardized in 2005 to support QoS for real time traffic. In addition to EDCA and HCCA explained in Section 1.3.5, it also supports Block ACK, where a block of frames are acknowledged with a BlockACK frame, and packet aggregation to improve the channel utilization. Most of the recent wireless cards support the features.
- 802.11f: It allows communication between APs through Inter Access Point Protocol (IAPP). IAPP was a trial-use recommendation and proposed to transfer the user context or user authentication information, but it was withdrawn in 2006.
- 802.11i: It was proposed in 2004 to support security in WLANs. The previous weak Wired Equivalent Privacy (WEP) was known to be weak and WiFi Protected Access (WPA) was proposed in WiFi Alliance and it was extended to WPA2 or Robust Security Network (RSN) in 802.11i.
- 802.11n draft: It supports a higher throughput using Multi-Input Multi-Output (MIMO). Current products using the Draft-N support 100 Mb/s and expected to support the higher data rate when it will be standardized in 2008.
- 802.11r draft: It was proposed to support fast roaming between APs with security enabled using the 802.11i standard, and it is expected to be standardized in 2008.
- 802.11k draft: Radio resource measurement enhancement. To avoid overload of an AP, which occurs when signal strength only is used for handoff decision, 802.11k will provide a better resource measurement method for better utilization of resources; the AP can ask clients to report the status of physical or MAC layer, and the clients can ask it to the AP also.

**Part I**

**QoS for User Mobility**

## Chapter 2

# Reducing MAC Layer Handoff Delay by Selective Scanning and Caching

When a wireless client moves out of the range of the current AP, it needs to find a new AP and associate with it. This process is called *layer 2 handoff* or *MAC layer handoff*. As this chapter will show, the MAC layer handoff takes too long for seamless VoIP communications, and therefore, I propose a novel and practical handoff algorithm that reduces the layer 2 handoff latency.

### 2.1 Standard layer 2 handoff

First, we investigate the standard layer 2 handoff and identify the problem.

#### 2.1.1 Layer 2 handoff procedure

When a client is moving away from the AP it is currently associated with, the signal strength and the signal-to-noise ratio (SNR) of the signal from the AP decrease. Generally, when they decrease below a threshold value, a handoff is triggered, even though other metric like the retry rate of data packets can be used. The handoff process can be divided into three logical steps: probing (scanning), authentication, association [51].

**Probing** can be accomplished either in passive or active mode. In passive scan mode, the client listens to the wireless medium for beacon frames, which provide a combination of timing and control information to clients. Using the information and the signal strength of beacon frames, the client selects an AP to join. During passive scanning, clients need to stay on each channel at least for one beacon interval to listen to beacons from all APs on the channel, and thus it takes long time to scan all channels; for example, when the beacon interval is 100 ms, it takes 1.1 s to scan all 11 channels in 802.11b, using passive scanning.

Active scanning involves transmission of probe request frames by the client in the wireless medium and processing of the received probe responses from the APs. The active scanning proceeds as follows [23]:

1. Clients broadcast a probe request to a channel.
2. Start a probe timer.

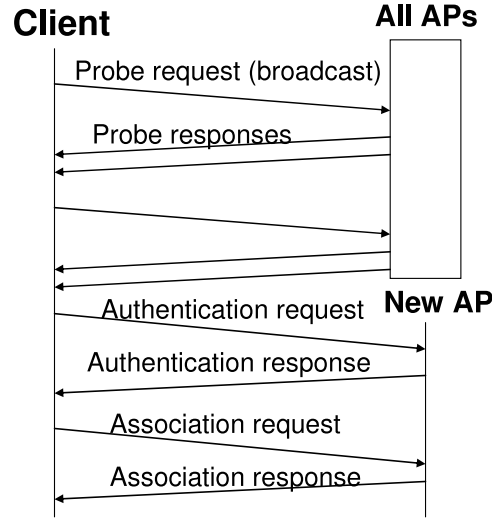


Figure 2.1: Layer 2 handoff process in IEEE 802.11

3. Listen for probe responses.
4. If no response has been received by *minChannelTime*, scan the next channel.
5. If one or more responses are received by *minChannelTime*, continue accepting probe responses until *maxChannelTime*.<sup>1</sup>
6. Move to the next channel and repeat the above steps.

After all channels have been scanned, all information received from the probe responses is processed so that the client can select which AP to join next.

While passive scanning has the advantage that clients can save power because they do not transmit any frames, active scanning is used in the most wireless cards because the passive scanning takes too long.

**Authentication** is a process by which the AP either accepts or rejects the identity of the client. The client begins the process by sending the authentication request frame, informing the AP of its identity; the AP responds with an authentication response, indicating acceptance or rejection.

**Association:** After successful authentication, the client sends a reassociation request to the new AP, which will then send a reassociation response to client, containing an acceptance or rejection notice.

Fig. 2.1 shows the sequence of messages expected during the handoff.

### 2.1.2 Layer 2 handoff time

According to many papers like [51] [31] [52] [37] [84] [71] and also the experiments in this study, the scanning delay dominates the handoff delay, constituting more than 90%. As Fig. 2.2 shows, the handoff time takes from 200 ms to 500 ms. The fluctuation comes from the number of APs that responded to the probe requests in each experiment because the client needs to wait for longer time (*maxChannelTime*) when any AP is detected on each channel. That is, the total handoff time can be represented as

<sup>1</sup>*minChannelTime* and *maxChannelTime* values are device dependent.

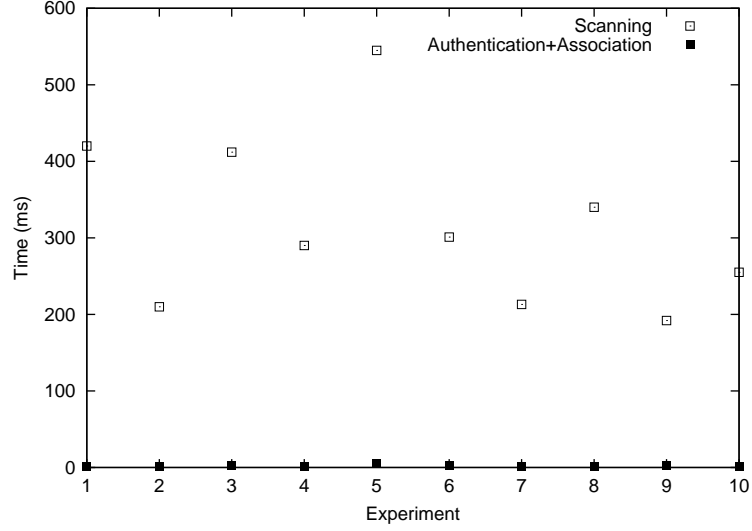


Figure 2.2: Layer 2 handoff time in IEEE 802.11b

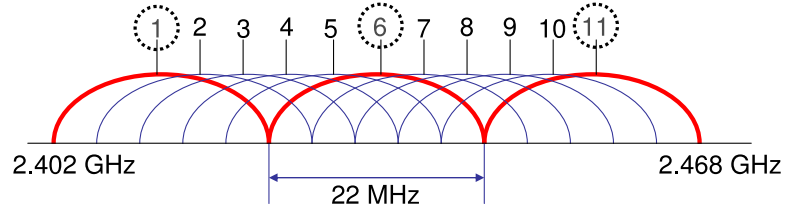


Figure 2.3: Channels used in IEEE 802.11b

Total handoff time =  $\maxChannelTime \times \text{number of channels on which APs responded} + \minChannelTime \times \text{number of unoccupied channels scanned} + \text{channel switching time} \times \text{total number of channels scanned}$

, and the number of APs responded varies according to channel status at the time of scanning, even in the same spot.

Also, we can see that scanning time takes most of the handoff time, and therefore, the key to reduce the layer 2 handoff delay is to reduce the scanning delay.

## 2.2 Fast layer 2 handoff algorithm

In order to reduce the probe delay, I propose a new handoff algorithm, Selective Scanning and Caching. Selective Scanning improves the handoff time by minimizing the number of channels to be scanned, and Caching reduces the handoff time significantly by eliminating scanning altogether when possible.



### 2.2.1 Selective Scanning

The basic idea of Selective Scanning is to reduce the number of channels to scan by learning from previous scanning. For example, if APs were found on channel 1, 6, and 11 and the client is associated with an AP on channel 11, then in the next handoff the probability that new APs will be found on channel 1 and 6 is very high, and it is not very likely that another AP will be found on channel 11 again because of the co-channel interference. Thus, the client should scan channel 1 and 6 first. Also, as shown in Fig. 2.3, among the 11 channels used in IEEE 802.11b standard, only three, channel 1, 6, and 11, do not overlap. Therefore, most of the APs are configured with these channels to avoid co-channel interference. The Selective Scanning algorithm is based on this idea.

In Selective Scanning, when a client scans the channel for APs, a channel mask is built. During the next handoff, this channel mask will be used in scanning. In doing so, only a well-selected subset of channels will be scanned, reducing the probe delay. The Selective Scanning algorithm is described below.

1. When the wireless card interface driver is first loaded, it performs a full scan, i.e., it sends out a Probe Request on all the channels and listens to responses from APs. Otherwise, scan the channels whose bits in the channel mask are set, and reset all the bits in the channel mask.
2. The new channel mask is created by turning on the bits for all the channels in which a Probe Response was heard as a result of step 1. In addition, bits for channel 1, 6, and 11 are also set, as these channels are more likely to be used by APs in 802.11b/g networks.
3. Select the best AP, for example, the one with the strongest signal strength from the scanned APs, and connect to that AP.
4. The channel the client connects to is removed from the channel mask by clearing the corresponding bit, as the possibility that adjacent APs are on the same channel as the current AP is small. Thus, the final formula for computing the new channel mask is 'scanned channels (from step 2)  $\oplus$  non-overlapping channels (1, 6, and 11 in 802.11b/g)  $\ominus$  the current channel'.
5. If no APs are discovered with the current channel mask, the channel mask is inverted and a new scan is done.

Fig. 2.4 shows the flowchart for the Selective Scanning algorithm.

### 2.2.2 Caching

By using Selective Scanning, clients need to typically scan only two channels (in 802.11b/g) to find a new AP, but the scan time can be reduced further by storing the scanned AP information in a AP cache at the client. The AP cache consists of a table with  $N$  entries, each of which contains the  $L$  MAC addresses of scanned APs, the MAC address of the current AP as the key, and the channel used by the APs. This list is automatically created or updated during handoffs.

Table 2.1 shows the cache structure. The length of an entry ( $L$ ) and the number of entries ( $N$ ) depend on the implementation and environment. Generally, a larger  $L$  increases the hit ratio but may increase the handoff time, and a larger  $N$  helps when clients are highly mobile. In the experiments in this study, the cache has a width of two ( $L = 2$ ), meaning that it can store up to two adjacent APs in the list.

The caching algorithm is described in detail below.

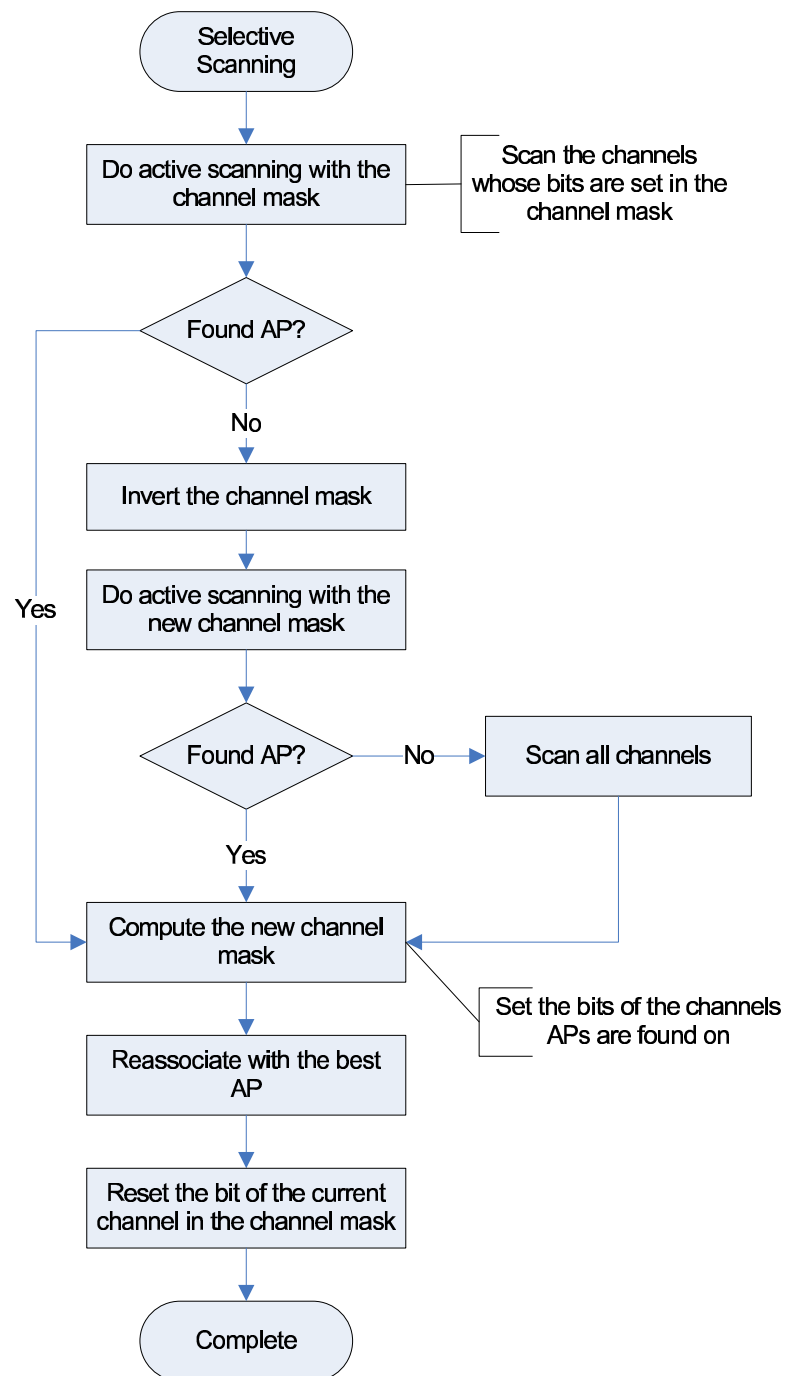


Figure 2.4: Selective scanning procedure

1	Key	AP 1	AP 2	...	(AP $L$ )
2	MAC1 (Ch1)	MAC2 (Ch2)	MAC3 (Ch3)	...	
...					
$N$					

Table 2.1: Cache structure

1. When a client associates with an AP it has not seen before, the AP is entered in the cache as a key. At this point, the list of AP entries, corresponding to this key, is empty.
2. When a handoff is needed, the client first searches the entries in cache corresponding to the current key.
3. If no entry is found (cache miss), the client performs a scan using the Selective Scanning algorithm described in Section 2.2.1. The best  $L$  results ordered based on signal strength or some other metric are then entered in the cache with the old AP as the key.
4. If an entry is found (cache hit), the client tries to associate with the first AP in the entry. If this succeeds, the handoff procedure is complete.
5. When the client fails to connect to the first AP in the cache, the next AP is tried. If the associations with all the APs in the cache entry fail, Selective Scanning starts.

From the above algorithm, we can see that scanning is required only if a cache miss occurs; every time we have a cache hit, no scanning is required.

Usually, using cache, it takes less than 5 ms to associate with the new AP. But, when the client fails to associate with the new AP, the wireless card waits for a longer time, up to 15 ms<sup>2</sup>. To reduce this time-to-failure, a timer is used. The timer expires after 6 ms, and the client will then try to associate with the next entry in cache. Thus, in the worst case, it takes up to  $L \times 6$  ms to start Selective Scanning in the worst case.

Other algorithms to improve the cache hit ratio can be added. However, the improvement would be minor because a cache miss does not significantly affect the handoff latency. As mentioned above, when the client fails to associate with an AP in the cache, the time-to-failure is only 6 ms, and thus if the client fails to associate with the first AP in the cache and associates with the second AP, the additional handoff delay is only 6 ms. When it fails to associate with both APs in the cache, the total handoff delay is 12 ms plus Selective Scanning time, all of this still resulting in a significant improvement compared to the original handoff time.

## 2.3 Implementation

Usually the handoff procedure is handled by the firmware in the wireless card, which we cannot modify. Thus, using the HostAP driver [46], the whole handoff process was emulated in the driver to implement the new handoff algorithm.

The HostAP driver is a Linux driver for wireless LAN cards based on Intersil's Prism2/2.5/3 802.11 chipset [46]. Wireless cards using these chipsets include the Linksys WPC11 PCMCIA card, the Linksys WMP11 PCI card, the ZoomAir 4105 PCMCIA card, and the D-Link DWL-650 PCMCIA card.

<sup>2</sup>Actual values measured using Prism2/2.5/3 chipset cards. These values may vary from chipset to chipset.

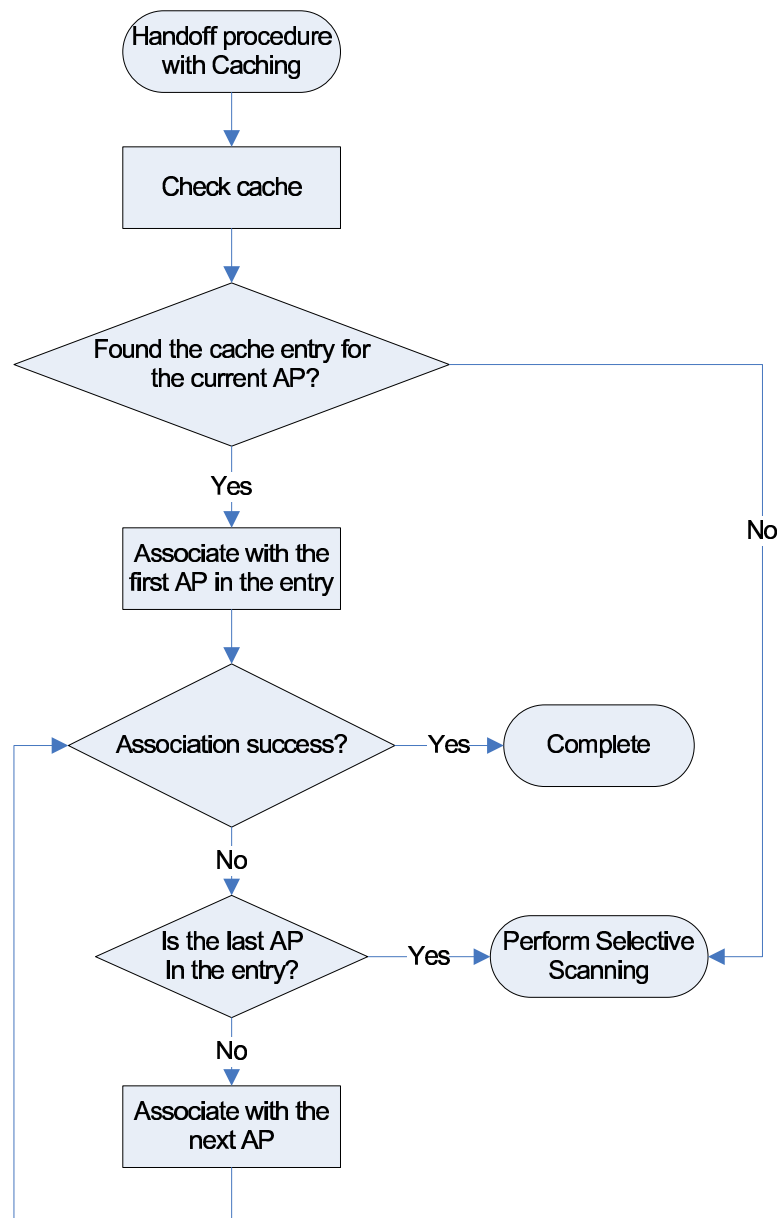


Figure 2.5: Caching procedure

The driver supports a so-called Host AP mode, i.e., it takes care of IEEE 802.11 management functions in the host computer and acts as an access point. This does not require any special firmware for the wireless LAN card. In addition to this, it supports normal station operations as a client in BSS and possible also in IBSS.<sup>3</sup>

The HostAP driver supports commands for scanning APs and associating with a specific AP. It is also possible to disable the firmware handoff by switching handoff mode to the manual mode where the HostAP driver can trigger handoffs. By using the manual handoff mode, it was possible to activate the handoff using the fast handoff algorithm in the driver.

## 2.4 Experiments

In the experiments, the total handoff time and the delay and packet loss caused by the handoff were measured, using the normal handoff algorithm and the Selective Scanning and Caching algorithm. This section describes the hardware and software for the measurements, the environment, and the experimental results.

### 2.4.1 Experimental setup

For the measurements, three laptops and one desktop were used. The laptops were a 1.2 GHz Intel Celeron with 256 MB of RAM running Red Hat Linux 8.0, a P-III with 256 MB of RAM running Red Hat 7.3, and another P-III with 256 MB RAM running Red Hat Linux 8.0. Linksys WPC11 version 3.0 PCMCIA wireless NICs were used in all three laptops. The desktop was an AMD Athlon XP 1700+ with 512 MB RAM running Windows XP. The 0.0.4 version of the HostAP driver was used for all three wireless cards, with one of them modified to implement the algorithms, and the other two cards were used for sniffing. Kismet 3.0.1 [36] was used for capturing the 802.11 management and data frames, and Ethereal 0.9.16 [14] was used to view the dump generated by Kismet and analyze the result.

### 2.4.2 Experimental environment

The experiments were conducted in the 802.11b wireless environment in the CEPSS building at Columbia University, on the 7th and the 8th floor, from Oct to Dec in 2003. With only two laptops running the sniffer (Kismet), many initial runs were first conducted to explore the wireless environment, specifically the channels of the APs and the places where handoffs were triggered.

The measurements for packet loss and delay were taken in the same space, but after some rogue APs were removed, from Jan to Feb in 2004. This change in the environment caused a reduction of the original handoff time and consequentially a drastic reduction of the packet loss. This will be shown in Section 2.4.4.

### 2.4.3 Measurement

One sniffer was set to always sniff on channel 1 (as the first Probe Request is always sent out on channel 1 in normal active scanning), and the other sniffer on the other channel the client was expected to associate to. For the measurement, the system clock of the three laptops was synchronized using the Network

---

<sup>3</sup>IBSS, also known as ad-hoc network, comprises of a set of stations which can communicate directly with each other, via the wireless medium, in a peer-to-peer fashion.

Experiment	1	2	3	4	5	6	7	8	9	10	avg
Original handoff	457	236	434	317	566	321	241	364	216	274	343
Selective Scanning	140	101	141	141	141	139	143	94	142	101	129
Caching	2	2	4	3	4	2	2	2	2	2	3

Table 2.2: Handoff delay (ms) in the experiments

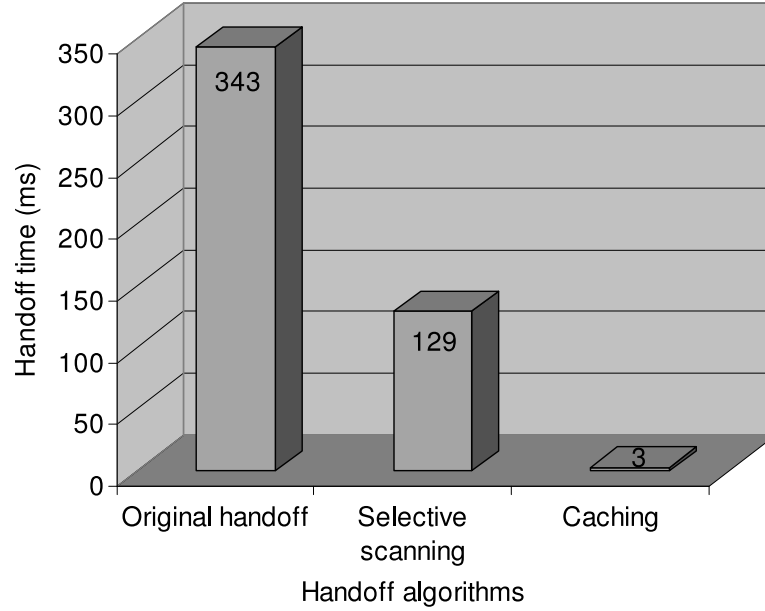


Figure 2.6: Layer 2 handoff time in 802.11b

Time Protocol (NTP). Also, to avoid multi-path delays, the wireless cards were kept as close as physically possible during the measurements.

For measuring the packet loss, in addition to the three laptops, the desktop was used as a sender and receiver. A UDP packet generator was used to send and receive data packets. Each UDP packet contained a packet sequence number in the data field to measure the packet loss.

#### 2.4.4 Experimental results

##### Handoff time

Table 2.2 shows the experimental results, and Fig. 2.6 shows the average. As can be seen, Selective Scanning improved the handoff time considerably, with an average reduction of 40%. But, even this reduced time is not good enough for seamless VoIP communication. However, using the cache, the handoff latency time drops to a few ms, which achieves seamless VoIP communication. This huge reduction was possible because scanning, which took more than 90% of the total handoff time, was eliminated by using Caching.

Also, we can notice that the handoff time using the original handoff algorithm fluctuates from

	w/o transmission (receiver)	with transmission (sender)
Original Handoff	182.5	201.5
Selective Scanning	102.1	141.1
Cache	4.5	3.9

Table 2.3: Handoff time in the environment without rogue APs (ms)

Experiment	1	2	3	4	5	6	7	8	9	10	avg
Original Handoff	281	229	230	210	209	227	185	174	189	168	210
Selective Scanning	185	132	147	131	204	182	164	133	151	184	161
Cache	0	0	0	0	0	0	0	0	0	0	0

Table 2.4: Packet delay (ms) during handoff in mobile sender

240 ms to 560 ms while that using Selective Scanning is relatively stable, varying only between 100 ms and 140 ms. This is because clients found rogue APs, which were configured with overlapping channels, by scanning all channels using the original handoff algorithm; when a client finds any AP on a channel, it needs to wait for longer time as explained in Section 2.1.1. Thus, the handoff time depends on the number of rogue APs scanned.

#### Handoff time with packet transmission

To measure the packet loss and delay caused from handoffs, UDP packets with 160 B packet size were transmitted during handoffs, simulating G.711 64 kb/s VoIP traffic. The environment was also slightly changed as mentioned in Section 2.4.2. The rogue APs have been removed, which decreased the scanning time significantly. As Table 2.3 shows, the handoff time (without packet transmission) using original handoff significantly decreased to 182 ms. It is because the client was able to scan other channels quickly by waiting for only *minChannelTime* because of the removal of rogue APs on overlapping channels; when rogue APs exist, they send the probe response frames to the probe request frames, and the client needs to wait longer in the channel, as explained in Section 2.1.1. Using the Selective Scanning algorithm, the change of handoff time is not significant, because clients do not scan the channels of rogue APs, using the channel mask. The slight decrease was because any AP(s) on channel 1, 6, or 11 were removed. Thus, we can notice that the behavior of the Selective Scanning algorithm is not dependent on the environment, while the original handoff performance is very much affected by it.

Table 2.3 also shows the handoff time using original and Selective Scanning algorithm with packet transmission is larger than that without packet transmission, which means that transmitting data packets during a handoff increases the handoff time. This is because data packets are transmitted *during* the handoff process, in particular between the last probe response and the authentication request.

#### Packet delay

The packet transmissions are delayed during handoff because the management frames have higher priority than data frames. Table 2.4 shows the average delay of packets sent during handoffs in each experiment. Even though some packets can be transmitted during the handoffs, we can see that the packet delay at the sender is almost as large as the handoff time.

Experiment	1	2	3	4	5	6	7	8	9	10	avg
Original Handoff	36	55	32	79	37	122	134	32	69	36	63
Selective Scanning	88	24	26	19	31	28	46	26	64	18	37
Cache	16	15	14	14	16	15	23	21	15	14	16

Table 2.5: The number of packets lost during handoff in mobile receiver

Experiment	1	2	3	4	5	6	7	8	9	10
Bridging delay (ms)	132	138	136	137	138	144	141	135	134	132

Table 2.6: Bridging delay

### Packet loss

When the UDP sender performs handoffs, no packet loss occurs during handoff and all packets are transmitted during and after handoffs with some additional delay. However, when the UDP receiver performs handoffs, packet loss can happen; if the AP sends the packets while the receiver is performing scanning, the packets are lost regardless of retransmissions. Table 2.4.4 shows the packet loss in the experiments. We can notice that too many packets are lost during the handoff, considering the handoff time and the packetization interval. For example, using Selective Scanning, the handoff time at the receiver was about 100 ms, and theoretically 5 to 6 packets can be lost during the handoff, for a 20 ms packetization interval. However, on average 37 packets are lost using Selective Scanning. The big difference is caused by *bridging delay*, which is the time needed for updating the MAC addresses to the Ethernet switches [51]. If the mobile client sends and receive packets simultaneously, the Ethernet switch will update the MAC address using the packets sent from the mobile client, and the bridging delay would disappear.

Table 2.6 shows the bridging delay, which is about 140 ms<sup>4</sup>. It was measured from the time between the association response frame and the first data packet from the new AP. Due to the bridging delay, when handoff happens and the client associates to the new AP, the switch continues to send the packets to the old AP in the old channel until the switch is updated, and the packets are lost in addition to the packets sent during the handoff.

As can be seen in Fig. 2.7, when the receiver is performing the handoff, the packet loss drops to about 60% and 40% using Selective Scanning and Caching, respectively. When using Caching, the effect of the bridging delay is particularly prominent; even though the handoff time is only a few milliseconds using Caching, the packet loss is still considerable. However, the packet loss would significantly decrease when clients transmit and receive packets at simultaneously, as mentioned earlier.

## 2.5 Related work

Prior to this effort, Arbaugh et. al. [51] measured the handoff time extensively using APs and wireless cards from various vendors and showed the discovery phase (scanning time) is the most time consuming part of the handoff process, taking over 90% of the total handoff delay, while (re)association time contributes only a few milliseconds. They have also shown that the handoff time varies significantly depending on the combination of models of wireless cards and the APs. However, they could not identify the specific reason. Also, they did not consider the effect of packet transmission on the handoff time and vice versa.

<sup>4</sup>Actual values may vary according to the environment.



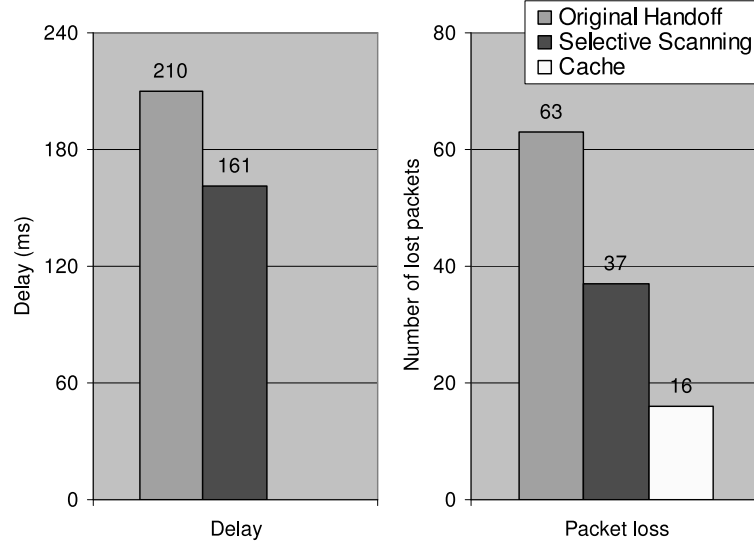


Figure 2.7: Packet loss at the mobile receiver and packet delay at the mobile sender

Kim et. al. [37] also proposed a selective scanning algorithm, where clients are notified of the neighbor AP information including the channels from a server that manages the neighbor graph of the APs, and clients scan only the APs in the graph and also wait only until the APs in the graph respond on each channel. However, their proposal relies on the use of neighbor graphs, and this approach requires changes in the network infrastructure, use of IAPP [25], and a central server. Also, a separate message exchange protocol is required to retrieve the neighbor graph from the server or APs.

This chapter focused on reducing scanning delay since scanning takes most of the handoff time when security is disabled. However, when security is enabled in handoffs, authentication and association take longer because clients and the AP need to exchange the security information. The following three papers studied the handoff with security enabled and tried to reduce the association and authentication delay.

Arunesh et. al. in [52] focused on reducing the reassociation delay. The reassociation delay is reduced by using a caching mechanism on the AP side. This caching mechanism is based on the IAPP protocol [25], which is used for the APs to transfer client context to other APs, in order to exchange the client context information between neighboring APs. The cache in the AP is built using the information contained in an IAPP Move-Notify message or in the reassociation request sent to the AP by the client. By exchanging the client context information with the old AP, the new AP does not require the client to send its context information in order to reassociate, hence reducing the reassociation delay.

Sangheun et. al. in [57] and Park et. al. in [58] focused on the IEEE 802.1x authentication process. This process is performed after the client has already associated with a new AP. The IEEE 802.1x authentication delay is reduced by using the Frequent Handoff Region (FHR) selection algorithm. After a client is associated with an AP, the FHR is computed using some handoff pattern factors, and the security context of the client is transmitted the APs in the region.

After the effort in this chapter, several new approaches have been proposed to improve the layer 2 handoff delay. Ramani et. al. [65] proposed SyncScan, which is based on passive scanning. The biggest

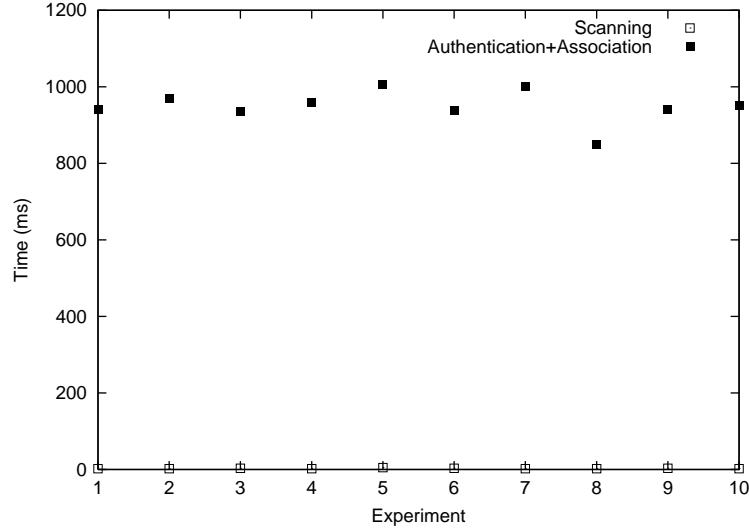


Figure 2.8: Handoff time in IEEE 802.11a

problem of passive scanning was that clients need to stay on a channel at least for the beacon interval to hear beacons from all APs on the channel, as mentioned in Chapter 2.1.1. In SyncScan, all the APs synchronize transmissions of their beacons so that clients can hear all beacons during the short amount of time. Also, to avoid the packet delay due to scanning in 802.11, a channel is scanned every few hundred milliseconds and data packets are transmitted in between. Thus, it takes long time to scan all channels. SyncScan allows clients to achieve seamless handoffs by eliminating scanning when handoffs are required, but it requires changes in all APs, which I wanted to avoid. Also, the long total scanning time (5.5 s with 500 ms scanning interval in 802.11b) is a problem because the scanned AP information might be too old when handoffs are required.

Wu et. al. [89] proposes Proactive Scan, which reduces the handoff time by decoupling the scanning from the handoff procedure. Proactive Scan scans a channel every few hundreds milliseconds during data communication before handoff is required. Even though it can reduce the handoff time, it takes too long time to scan all channels and the scanned AP data could be too old when handoff is required, as in the SyncScan approach. Thus, to reduce the scanning time, they filter the channels to scan according to the priorities of channels, which is very similar with the Selective Scanning algorithm. Also, they consider the asymmetry between uplink and downlink quality to improve handoff decision; clients check the uplink and downlink data rate for handoff decision, but it is not a critical problem in handoff.

## 2.6 Conclusion

Layer 2 handoff occurs very frequently while wireless clients move around, in particular, in buildings because the coverage of an AP is very limited. The layer 2 handoff time takes up to 500 ms, and it reduces the QoS of VoIP service. In this chapter, a fast handoff algorithm using Selective Scanning and Caching was described.

I have implemented the algorithms using the HostAP driver and showed via experiments that the handoff delay decreases to about 130 ms by only using the Selective Scanning algorithm and to 3 ms

by using Caching. This reduction in handoff latency also considerably decreased packet loss and packet delay. Also, the new handoff algorithm can be implemented by modifying only the wireless card drivers of clients, without changing the firmware or the APs, while other previous approaches require changes in the standard or infrastructure like APs.

Another important result of this study is that by using Selective Scanning and Caching, the probing process, the most power consuming phase in active scanning, is reduced to the minimum. This makes it possible to use the active scanning procedure also in those devices such as PDAs where power consumption is a critical issue.

Also, the algorithm can be used to reduce the handoff time in IEEE 802.11g networks because it uses the same channels as 802.11b. In IEEE 802.11a, we can improve the handoff delay significantly using the Selective Scanning and Caching. Fig. 2.8 shows the original handoff time in IEEE 802.11a networks. As can be seen, the discovery phase is still the most time consuming phase of the handoff process, and the total handoff time takes more than a second. It is because there are more channels (more than 24 channels depending on countries) are available in 802.11a, even though only 12 channels are non-overlapping.

## Chapter 3

# Reducing IP Layer Handoff Delay by Fast Subnet Detection and Temporary IP address

### 3.1 Introduction

IP layer handoff or layer 3 (L3) handoff happens, when a wireless client moves from a subnet to a different subnet by layer 2 handoff. Two of the main problems encountered in a L3 handoff process are the detection of subnet change and the long IP address acquisition time via DHCP [12].

For the first problem, subnet change detection, router advertisement can be used, but it takes too long time because different networks might use different intervals for transmitting router advertisements and these intervals can be very long, up to several minutes. Also, we cannot use ESSID for subnet change detection. Most large-scale 802.11 wireless networks use the same SSID everywhere. SSIDs are assigned according to administrative needs and not according to the topology of the wireless network, as explained in Section 1.3.1. Thus, this chapter introduces a fast subnet discovery mechanism using a DHCP query.

The second problem is that it takes up to several seconds to acquire a new IP address from DHCP servers [2]. In particular, the largest component of the DHCP assignment procedure is the time between the DHCP DISCOVER message sent by the client and the DHCP OFFER message sent by the DHCP server. This problem will be described in the next chapter in detail. During this time, Duplicate Address Detection (DAD) is performed to be sure that the address the DHCP server wants to offer is not already used by some other clients.

In this chapter, I introduce a client side solution, the concept of a temporary IP address that can be used by the client while waiting for the DHCP server to assign it a new IP address. A server side solution which improves the DAD procedure, called passive DAD (pDAD), will be introduced in the next chapter.

### 3.2 Layer 3 handoff algorithm

Fig. 3.1 shows the complete fast layer 3 handoff procedure starting with layer 2 handoff; each step will be described in detail in this section.

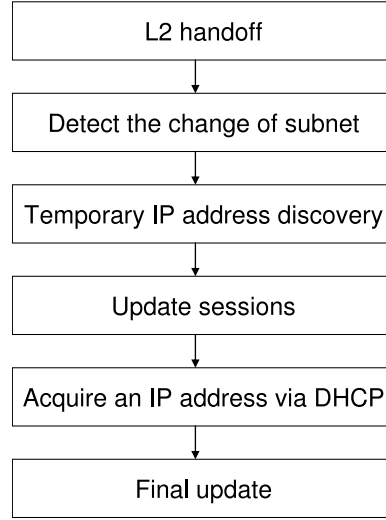


Figure 3.1: Fast layer 3 handoff procedure

1	Key	AP 1	AP 2	...	(AP $L$ )
2	MAC,Ch,SID	MAC1,Ch1,SID1	MAC2,Ch2,SID2	...	
...					
$N$					

Figure 3.2: Enhanced cache structure

### 3.2.1 Fast subnet change detection

The fast subnet change detection relies on the observation that each subnet is served by different DHCP servers or relay agents. Relay agents are used for the DHCP server to identify from which subnet a DHCP packet is coming when more than one subnet is present in a network [12]. This allows the DHCP server to assign a valid IP address to a client in its subnet. If the network has one subnet only, then there is no need for relay agents, and DHCP packets will be handled by the DHCP server directly.

After a layer 2 handoff is done, the client sends a DHCP REQUEST packet with the loopback address to the DHCP server. The DHCP server responds with a DHCP NACK packet, which contains the IP address of the relay agent, or the DHCP server itself if there is only one subnet under the DHCP server of the subnet the client is currently connected to. Generally, the DHCP server sends back the DHCP NACK packet quickly because the requested IP address is not valid, and it takes only one round trip time of a packet to detect the subnet change. Clients can detect the subnet change by comparing the IP address with the one in the previous subnet. If the client is in the same subnet, no further action is needed as it has performed a normal L2 handoff. However, if the client is in a different subnet, it has to initiate the L3 handoff process.

Also, in order to store the discovered subnet information, the cache mechanism, which was used for seamless layer 2 handoff [71] and described in Chapter 2 is improved. The structure of the enhanced cache is shown in Fig. 3.2, and now a subnet ID (SID) for each AP is stored in the cache. The IP address of the DHCP server or relay agent is used as the subnet ID. Once the client discovers a new subnet, it saves this information in the cache so that the next time it connects to the same AP, it will already know in which

subnet it is, and no subnet discovery process is necessary.

Therefore, when a client performs a L2 handoff and connects to a new AP, it has to check if a subnet change has occurred or not, by checking the L2 cache. If it has a valid value in the subnet ID field for the new AP, the client compares this value with the subnet ID value of the previous AP, and if the two fields have the same value, the subnet has not changed. Otherwise, the subnet has changed, and the client has to initiate the L3 handoff process. In this case, the L3 handoff process does not include a subnet discovery phase since the L2 cache already has the information. On the other hand, if it cannot find a valid value in the subnet ID field of the new AP, it has to initiate the subnet discovery procedure explained above.

According to the subnet information in the cache, the following three scenarios are possible, and the L3 handoff process changes according to the scenarios:

- Scenario 1: The client enters a new subnet for the first time ever.
- Scenario 2: The client enters a new subnet it has visited before, and it has an expired IP address lease for that subnet.
- Scenario 3: The client enters a new subnet it has visited before, and it still has a valid IP address lease for that subnet.

### 3.2.2 Discovering the temporary IP address

The basic idea is that clients scan a range of IP addresses to find a temporary IP address that can be used by the client while waiting for the DHCP server to assign it a new IP address. The temporary IP address selection procedure follows some heuristics based on a particular behavior of the DHCP server implementation; after the DHCP server has assigned all the IP addresses of its pool at least once, it will assign addresses to new clients based on an aging mechanism. The IP address that has not been assigned for the longest time will be assigned first. After some time, the way the IP addresses are allocated by the DHCP server is completely random, one with an exception that for any given client the DHCP server will try first to assign the last address that the client used earlier. Because of this randomness in assigning IP addresses, we can find a temporary IP address quickly by scanning a range of IP addresses in the subnet. In order to verify the randomness via experiments, the average number of consecutive IP addresses in use was measured in a wireless subnet. In the experiments, the number of consecutive IP addresses used at peak time has a 99th percentile value of 5. This means that in 99% of the cases we will have at most 5 consecutive used IP addresses before finding an unused one, a temporary IP address. However, the number of IP addresses occupied consecutively is not important because multiple IP addresses can be scanned in parallel. Thus, even in the wireless network where IP address utilization is high, we can find an unused IP address without incurring additional overhead.

To scan the unused IP addresses, the client sends an Address Resolution Protocol (ARP) [62] request packet to each candidate IP address. If any device is using the IP address, it should send an ARP response packet to the client. If the IP address is not used, there will be no answer for the request. Thus, if the client does not receive any response after a certain amount of time, the IP address can be used as a temporary IP address. ICMP ECHO also can be used for scanning, but it is not used because recently many firewall applications block incoming ICMP ECHO requests.

Here, the ARP response waiting time is very important. In the experiments, an ARP timeout value of 130 ms was used. As will be explained in Section 3.4.2, this value represents the 90th percentile of the total waiting time in the worst case scenario. The ARP timeout value must be chosen carefully because a

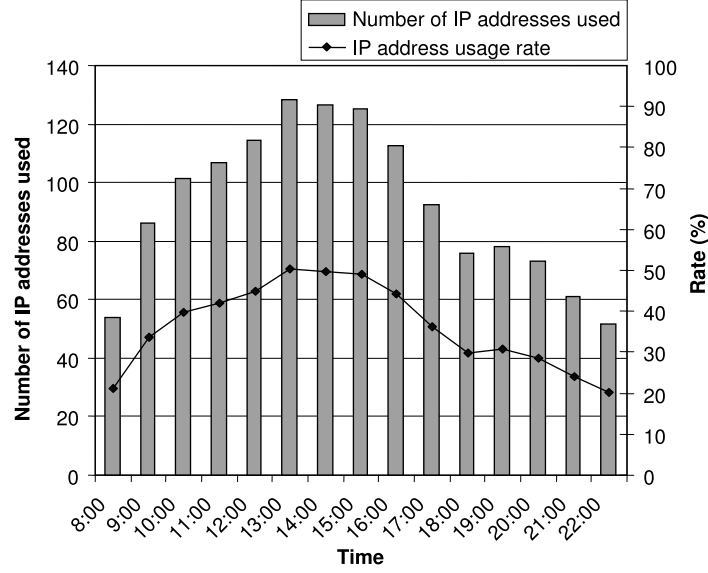


Figure 3.3: Average IP usage in the Columbia University wireless network

bigger value will increase the total handoff time, while a smaller value will introduce a risk for assigning a duplicate address.

Also, to identify the probability of choosing a duplicate address as a temporary IP address, the IP address usage rate was measured during weekdays in the wireless network of Columbia University, which is a representative large scale wireless network. During the peak time in the experiments, the IP address usage rate was about 50% (Fig. 3.3). By choosing the 90th percentile of the waiting time, the risk of picking an IP address currently in use as a temporary IP address even at the peak time, is only about 5%. Thus, the possibility of choosing a duplicate IP address as a temporary IP address is practically is very low.

In order to find a suitable temporary IP address for the new subnet, we select an IP address in random offset from the router IP address, which is usually the lowest one in the subnet. We then start sending ARP requests in parallel to 10 IP addresses selected in a sequence starting from the random IP address selected before. This will secure us with a temporary IP address since the probability of finding 10 consecutive IP addresses in use is very low, according to the experiments. In a busy wireless network, where the IP address utilization is very high, the larger number of IP addresses for search can be used, which does not increase the network traffic much considering that the small packet size of the ARP packets and the very low probability that more than one client performs a L3 handoff at any given time.

Another reason that we can find the temporary IP address easily is that abandoned IP address also can be used as the temporary IP address. In a wireless environment, we can safely assume that the degree of mobility of clients is high, and clients leave the subnet before leases of their IP addresses have expired. This means that usually there will be many IP addresses whose leases have not expired but that are not used and cannot be assigned to new clients. Using the temporary IP address scanning mechanism, these IP addresses can be detected and used as a temporary IP address.

In the second scenario in Section 3.2.1, where clients enter a new subnet they have visited before but the IP address lease for that subnet has expired, the temporary IP address is selected as described above. The only difference is that instead of sending ARP requests starting from a completely random IP address,

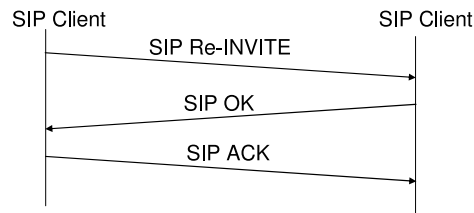


Figure 3.4: SIP session update

clients start from the IP address the clients had the last time they were in this subnet. In general, the DHCP server always tries to assign to a client the same IP address it assigned to the client last time. This makes the IP address clients last used in that subnet the perfect candidate for a temporary IP address, and perhaps the DHCP server will assign that same IP address as well.

In the third scenario, where clients enter a new subnet they have visited before with a valid IP address lease for that subnet, there is no need for a temporary IP address since clients still have a valid lease for the new subnet. In this case, clients can start using the IP address with the valid lease right away and send a DHCP REQUEST packet to the DHCP server in order to renew the lease.

### 3.2.3 Session updates

If there is any sessions, we need to update the sessions, and the process is application dependent.

#### SIP session update

Once a client has a valid IP address to use, the client can initiate a application layer handoff. In our experiments, Session Initiation Protocol (SIP) [68] was used as the application layer protocol. SIP is a text-based protocol for IP telephony, conferencing, and instant messaging. To initiate the call, an INVITE request is sent to a callee, and the callee can accept the call by sending a “200 OK” response. When the caller receives the OK message, it sends an ACK message and establishes a VoIP session. If the media information such as IP addresses changes, they need to re-establish the session by exchanging the above three messages; this process is called re-INVITE.

Thus, after the acquisition of a temporary IP address, the client needs to send a re-INVITE to the corresponding node (CN) informing the node of the change in IP address. When the node replies with an OK, the data exchange can be resumed. Note that the data exchange can be resumed after receiving the OK before receiving the ACK. The full sequence of signals exchanged is shown in Fig. 3.4.

Note that in scenarios one and two, only the correspondent node is aware of the temporary IP address, not the SIP home proxy server. SIP home proxy (or registration) server is generally used to store and query the contact information of users, the IP address of the devices of users are currently using. Thus, new sessions will not be accepted and/or initiated during the short interval when the client is using the temporary IP address.

#### IP address acquisition via DHCP

In scenarios one and two, the client has to request a new IP address from the DHCP server. This will not cause any interruption because the client is using the temporary IP address while waiting for the new IP



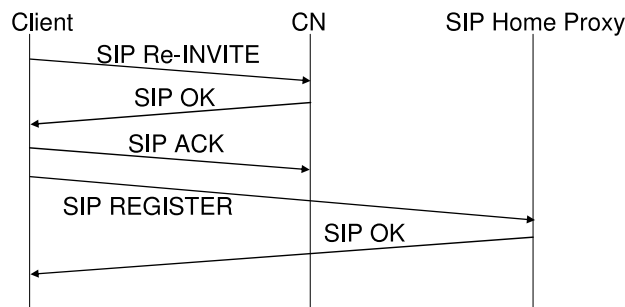


Figure 3.5: Full SIP session update

address. Also, in scenario three, this step is not required because the client already has an IP address with a valid lease that it can use for the particular subnet it moved into.

### Final update

As a final step, a new session update is required via a re-INVITE message exchange using the new IP address. Also, the client needs to register the new IP address with the SIP home proxy server so that new sessions can be accepted. Thus, a REGISTER request is sent to the SIP home proxy with the new IP address (Fig. 3.5).

Once the SIP session update has finished, we can then safely remove the temporary IP address and start using the new IP address assigned by the DHCP server. The switching between the temporary IP address and the new IP address is completely seamless.

The full handoff process for scenario one is shown in Fig. 3.6, including the subnet discovery phase. Note that the sequence of messages exchanged in scenario two and three is a subset of the messages exchanged in scenario one. The gray area shows the network connectivity disruption duration, and we can see that the network is disrupted only from the line of the L2 handoff to the first SIP session update.

## 3.3 Implementation

To implement the fast L3 handoff algorithm, a DHCP client, a wireless card driver, and a SIP client were modified. Linux (RedHat 9.0) was used as a platform because the source code of a DHCP client and a wireless card driver are open. Dhcp-pl2 [28] was used as a DHCP client, HostAP driver (hostap-0.0.4) [46] as a wireless card driver, and the *mca* from SIPquest (currently FirstHand Technologies [74]) as a SIP client.

Fig. 3.7 shows the architecture of the fast layer 3 handoff implementation. The communications among three components are newly implemented for the handoff algorithm. The wireless card driver communicates with the DHCP client via sockets to inform the completion of the layer 2 handoff and the subnet ID of the new AP so that the DHCP client can start a layer 3 handoff if necessary. The DHCP client communicates with the SIP client through IPC (Inter-Process Communication) using socket to initiate the application layer handoff (SIP re-INVITE) after the acquisition of a temporary IP address and a new IP address.

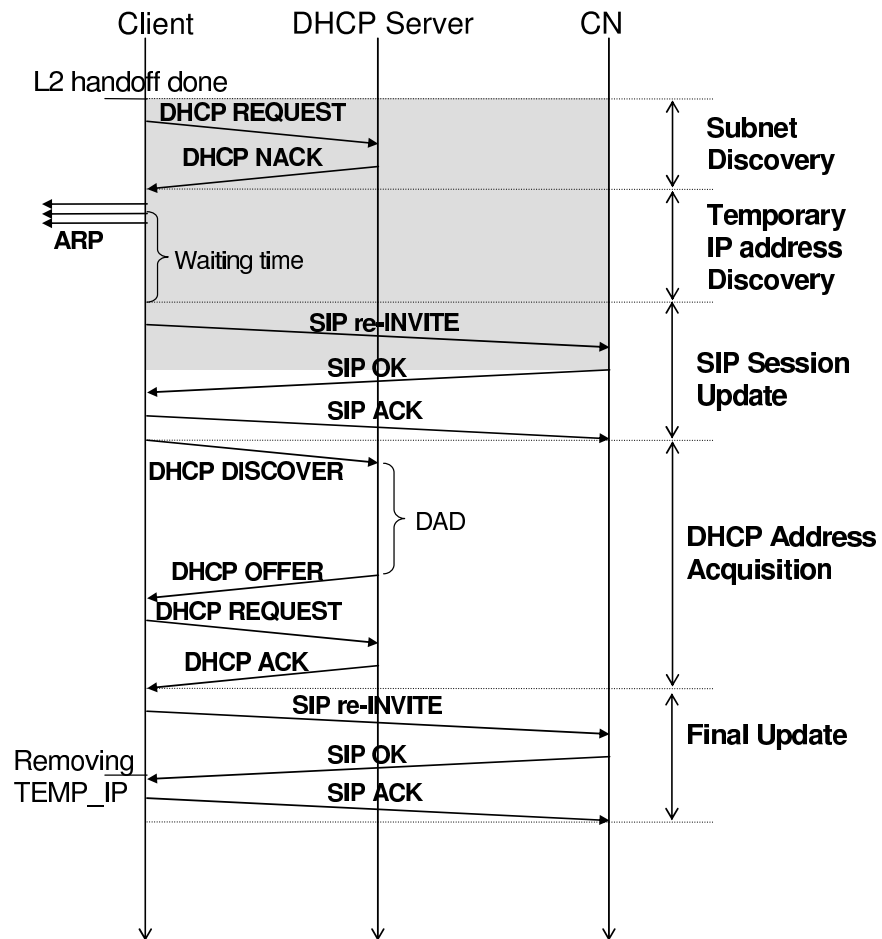


Figure 3.6: Full layer 3 handoff under scenario one (no lease)

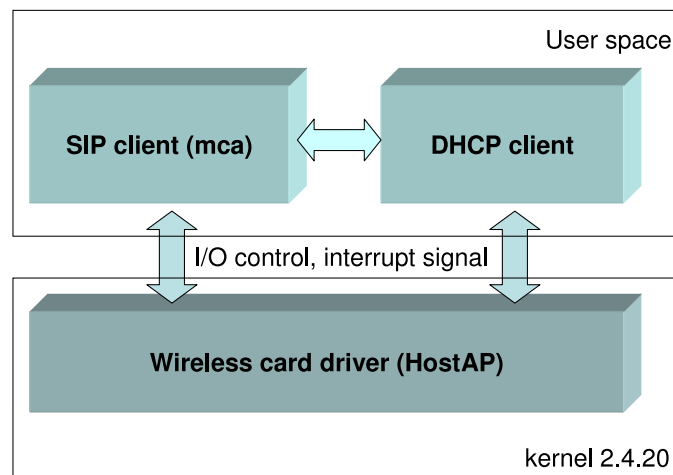


Figure 3.7: The architecture of the fast L3 handoff implementation

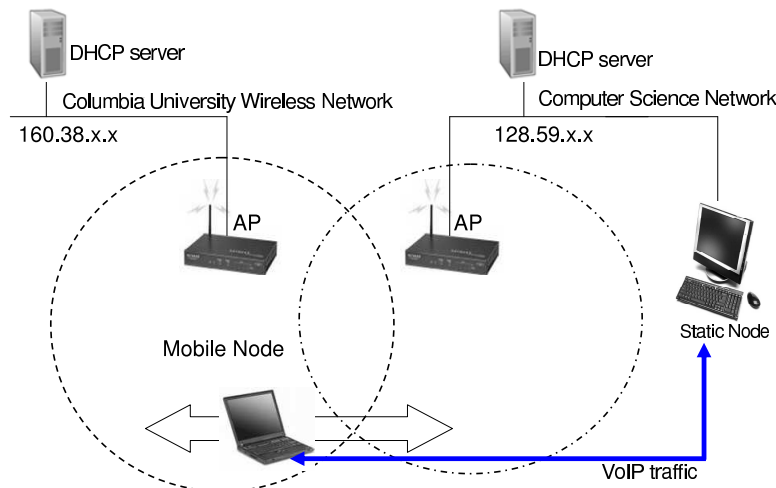


Figure 3.8: Experimental environment

## 3.4 Experiments

### 3.4.1 Test environment

Experiments were performed on the 7th floor of the CEPSPR Building at Columbia University from Feb to Apr in 2005. Since the Columbia University wireless network consists of only one subnet (160.38.x.x), an extra AP was set up to add the second subnet (128.59.x.x), which is Computer Science network (Fig. 3.8).

SIPc [80], a SIP client developed at Columbia University, was used as a fixed client on a Pentium IV 2.4GHz desktop running Windows XP. As a mobile client, a modified version of the SIP client from SIPquest was used in an IBM ThinkPad Pentium III with RedHat 9.0 (kernel 2.4.20).

### 3.4.2 Parameter calculation

Clients send ARP request packets to the subnet to find a temporary IP address, and the waiting time for the ARP responses is critical for the handoff time, as mentioned in Section 3.2.2. Thus, experiments to measure an optimal waiting time value for ARP responses was performed in the Columbia University wireless network, which is large enough to represent a large scale and busy wireless network environment. In the experiments, ARP requests were sent to the IP addresses from 168.38.244.1 to 168.38.246.255, and the response times were measured. In order to check the worst case scenario, the experiments were performed during the time of maximum network congestion (between 3:00pm and 4:00pm). According to the experimental results, the 90th percentile value of the ARP response time for detecting an IP address as in use, was 130 ms, and the 99th percentile value was 260 ms.

### 3.4.3 Measurements

Theoretically, the L3 handoff time is the time from the L2 association response frame to acquisition of the new IP address. However, in SIP, after getting an IP address, the mobile node needs to announce its new IP address to the CN. The SIP session update is also called the application layer handoff. However, because voice communication will be disrupted until the CN updates its session with the new IP address,

Table 3.1: IP address acquisition time in normal DHCP and the new approach

	Normal DHCP	Using Temp_IP
Lease has expired	518 ms	138 ms
Lease has not expired	7.5 ms	1 ms

the L3 handoff time is defined as the time from the association response frame to the SIP OK message in the first SIP session update. When computing the L3 handoff time, the time to acquire a new IP address using DHCP and the second SIP session update are not included because the network connectivity is not disrupted during that time.

In order to measure the L3 handoff time, association response frames and the SIP OK response need to be captured. To capture all the packets from and to the mobile node including the association response frames, Kismet [36] was used as the wireless sniffer. To capture all the SIP messages in the fixed node, Ethereal [14] was used.

Also, in the experiments, the packet loss during the L3 handoff was measured. The packet loss is defined as the number of packets sent from the CN between the association response frame and the SIP OK message, according to the definition of L3 handoff given earlier. All the nodes including the sniffer were synchronized using NTP [50].

#### 3.4.4 Experimental results

The L3 handoff time can be divided into four components: subnet detection time, IP acquisition time, client processing time, and SIP signaling time required for updating SIP session. The definition of each component is as follows:

**Subnet detection time:** The subnet detection time starts when the association response frame was sent from the AP and ends when the DHCP server sends the DHCP NACK frame for the DHCP REQUEST from client.

**IP address acquisition time:** Time from sending the first ARP request to expiration of the ARP response waiting timer.

**SIP signaling time:** From the INVITE message the client sent to the “200 OK” message the client received.

**Client processing time:** the time between IP address acquisition time and SIP signaling time.

The whole L3 handoff time was measured under the three scenarios specified in Section 3.2.1, the average of each component was taken, and the total L3 handoff time of each scenario was computed using the components.

##### IP address acquisition time

According to the DHCP specification [12], when the DHCP client needs to get an IP address, the client checks the lease file, which contains IP addresses, their lease time, and the subnet information. If the lease file has the IP address information of the new subnet and the lease is still valid, the client sends a DHCP

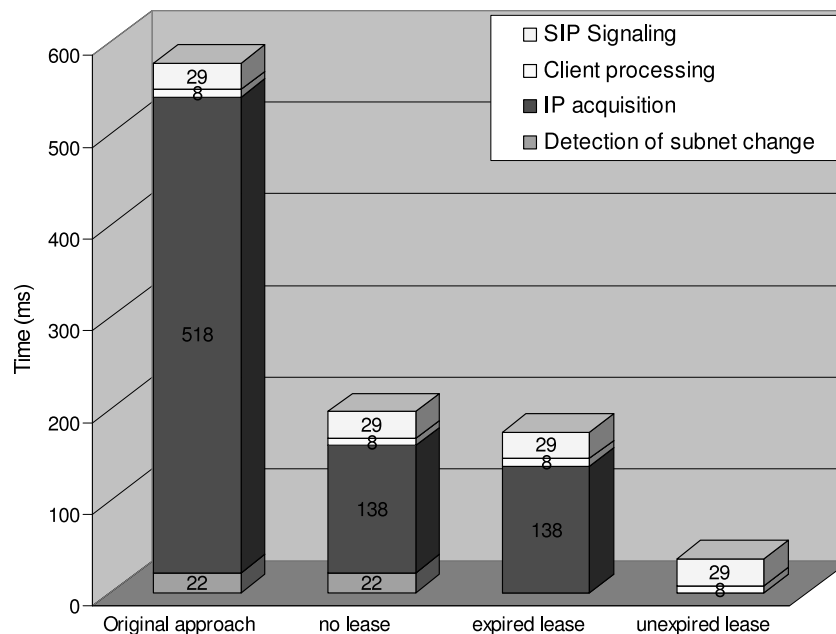


Figure 3.9: L3 handoff time using new approach

REQUEST packet to renew the leased IP address. Otherwise, if the lease of the IP address has expired or the file does not have any information about the subnet, it sends a DHCP DISCOVER packet to get a new IP address. Therefore, the IP address acquisition time was measured in each case and compared.

Table 3.1 presents the average IP address acquisition time for the standard DHCP procedure and for the new approach, in each case. We can see that when the lease has expired or does not exist, it took more than 500 ms to get a new IP address using the DHCP procedure due to duplicate address detection, while it took only about 138 ms (it is because of 130 ms of ARP response waiting time, which can be optimized depending on environments). Actually, the standard DHCP client implementation from Internet System Consortium (ISC) should use a one second waiting time for an ICMP response, but the waiting time changes randomly from 10 ms to 900 ms because of an architectural flaw in the implementation. Thus, the IP address acquisition time using DHCP would be more than 1 second without the implementation flaw.

Also, when the lease has not yet expired, it took 7 ms on average to renew it in the standard DHCP client, while it took 1 ms using the new approach. This is because the standard DHCP client binds to the IP address *after* it receives a DHCP ACK from the DHCP server, while the client using the new approach first binds to the leased IP address, and then it starts the process for renewing it.

### Total layer 3 handoff time

Fig. 3.9 shows the total L3 handoff time using each component. In the original approach, there is no standard way to detect the subnet change, and thus, the Linux client did not initiate L3 handoff, and Windows XP did but took more than 1 min. However, for the comparison, it is assumed in the original approach that the fast subnet detection mechanism described in Section 3.2.1 is used together with the DHCP process to acquire a new IP address, and the L3 handoff time still takes 550 ms due to the long IP

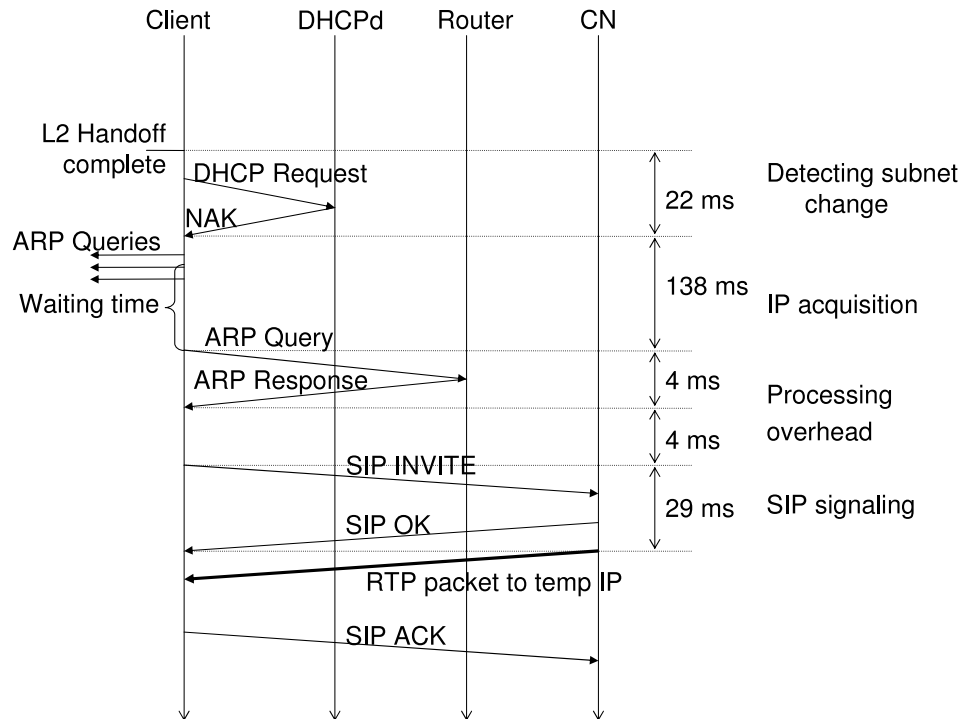


Figure 3.10: Messages exchanged during L3 handoff and delay components

Table 3.2: Packet loss during L3 handoff using the new approach

	No lease	Expired lease	Unexpired lease
Number of packets lost	13	12	3

address acquisition time. We can see that in the worst case (scenario 1), we can reduce the L3 handoff to a third of the original approach, using the new approach.

Fig. 3.10 presents all the components of the L3 handoff time in the worst case (scenario 1).

### Packet loss

Table 3.2 shows the packet loss during L3 handoff. We used 64 kb/s VoIP traffic with 20 ms packetization interval for the experiment. We can see that the number of packets lost is roughly proportional to the L3 handoff time, as expected.

## 3.5 Related work

A lot of work has been done on reducing the L3 handoff delay; however, very little has been done on reducing the DHCP acquisition time itself. Also, most of them require changes in the infrastructure or introduce new components, which the new approach in this study wanted to avoid.

Kim et. al. [39] try to reduce the L3 handoff delay by proactively reserving the new IP address for the new subnet while still in the old subnet. In particular, they acquire a new IP address and update the SIP session with the new address before performing the L2 handoff. Unfortunately, this approach requires changes to the DHCP protocol and to the network infrastructure as well because the IP request message needs to be sent from the old subnet to the new subnet.

The Dynamic Registration and Configuration Protocol (DRCP) [47] is a new protocol intended to replace DHCP. DRCP reduces the use of broadcast messages in a transaction, and the message size for limited wireless bandwidth. DRCP reduces the address allocation time allowing handoff times in the order of a few hundred milliseconds [39], still too large for real time applications. This new protocol would also require upgrading the entire network in order to be supported.

Akhtar et. al. [2] compare the L3 handoff delay of two different approaches, namely, SIP/DHCP and SIP/Cellular-IP. SIP is used for macro-mobility while DHCP and Cellular-IP are used for micro-mobility. They show how the SIP/Cellular-IP approach introduces a delay of about 0.5 seconds while the SIP/DHCP approach introduces, in the worst case scenario, a delay of about 30 seconds. The authors also show how most of the delay introduced in the second approach is due to the DHCP procedure. In any event, both approaches are unsuitable for real time applications.

Vali et. al. [82] introduce Hierarchical Mobile SIP (HMSIP) for micro-mobility of clients. A new component, called HMSIP agent, is installed as a local SIP registrar in every domain, and every mobile node registers with a HMSIP agent. When the IP address changes, it needs to update the session to HMSIP agent. This approach ignores the break during IP address acquisition time, and the new component should be installed in every visited network.

Dutta et. al. [13] propose three methods for reducing application layer handoff time. The first one places an RTP translator in every visited network. When a client gets a new IP address, it registers the new IP address with the SIP registrar of the visited network; then, the SIP registrar asks the RTP translator to forward the traffic associated with the old IP address to the new IP address. Another approach uses a Back-to-back User Agent (B2BUA). There are two B2BUAs in the middle of mobile host (MH) and correspondent host (CH), and when the IP address of the MH changes, MH just needs to update session to the B2BUA. The last approach uses a multicast IP address. When a client predicts a subnet change, it informs the visited registrar or B2BUA of a temporary multicast address as its contact or media address. Once the client arrives at the new subnet and gets a new IP address, it updates the registrar or B2BUA with the new unicast IP address. However, in both the first two methods, the time to acquire the new IP address is ignored.

## 3.6 Conclusion

In this chapter, a novel L3 handoff approach was introduced. In the approach, to detect subnet changes, clients send a DHCP REQUEST packet containing an invalid IP address which will cause the DHCP server to send a DHCP NACK packet. Clients then extract the relay agent or DHCP server IP address from the DHCP NACK frame and use it as a subnet ID to detect the subnet change. A temporary IP address is selected by sending ARP requests to a range of IP addresses to find an unused IP address. The temporary IP address will be used until a DHCP server assigns a new IP address to the client. In such a scenario, the L3 handoff takes about 190 ms, including the session update. Even though this does not make the handoff seamless, it represents a big improvement considering that the current Linux kernel does not support L3 handoff and that such a delay is more than 1 minute in Windows XP. When a client has already visited the new subnet once before and the lease for such subnet has not yet expired, the client can update its SIP

session with the IP address first and renew the lease later, achieving a seamless handoff with the delay of about 30 ms.

One of the goals of this study was to not change any infrastructure. All the changes required by the new approach are introduced on the client side. Only mobile nodes, namely, the wireless card driver and DHCP client, need to be modified, and this makes the solution more practical.

However, not introducing changes on the infrastructure side forced to introduce some tradeoffs between the total handoff delay and the duplicate address probability, even though it is very low. Therefore, the next chapter introduces a server side solution for the long IP address acquisition time, pDAD [15], which eliminates the time consuming DAD procedure at DHCP server so that the server can quickly assign new IP addresses to clients.



# Chapter 4

## Passive Duplicate Address Detection for DHCP

### 4.1 Introduction

As explained in Chapter 3, the largest contributor to the layer 3 handoff delay is the IP address acquisition. A client side solution using a temporary IP address was proposed in the last chapter, but the temporary IP address approach has to make a trade-off between the IP address acquisition time and the possibility of duplicate IP address, despite of the advantage that it works in any network only with changes on the client side. If changes in the infrastructure are allowed, the layer 3 handoff time can be further reduced without such a trade-off. Therefore, a server side solution, Passive Duplicate Address Detection (pDAD), is introduced in this chapter.

Fig. 4.1 shows the basic flow of the pDAD. pDAD is a framework that monitors the network and detects IP addresses currently in use in one or more subnets, and it collects information on which IP addresses are in use in a specific subnet and informs the DHCP server of such addresses. In doing so, the DHCP server already knows which addresses are in use when a client requests a new address and therefore it can assign the new address immediately without performing any further action during the assignment process. This allows us to remove any delay caused by DAD at DHCP servers during the IP address acquisition time.

It does not appear to be any directly related work to improve the DAD procedure at DHCP servers.

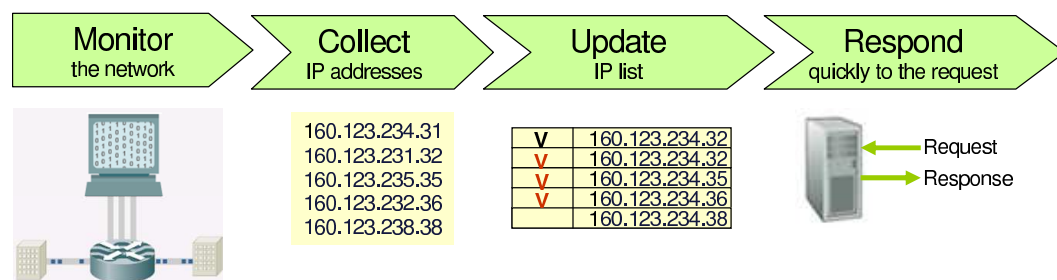


Figure 4.1: Outline of Passive DAD operation

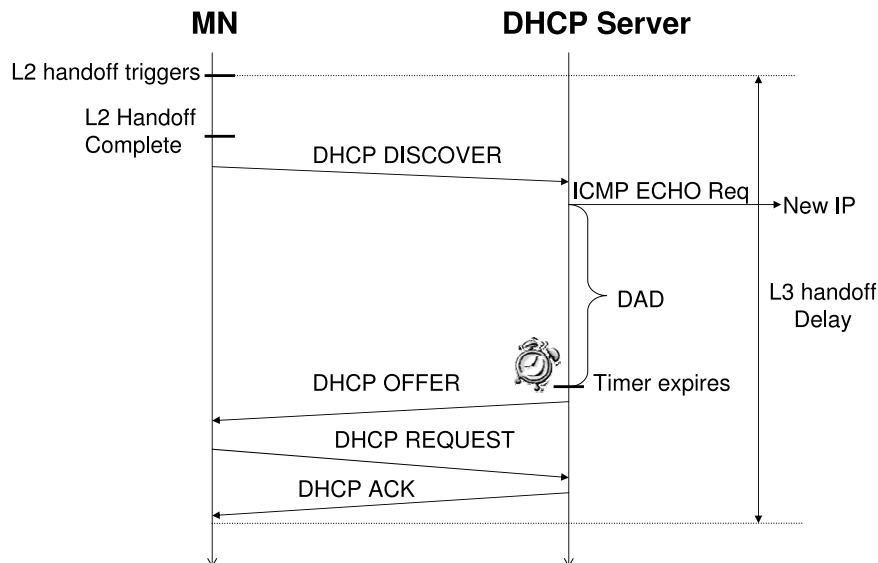


Figure 4.2: DHCP procedure

Most of the work done in the network community for optimizing DAD addresses DAD in the particular case of self-configuring networks such as ad-hoc networks [81] [87]. Other work has been done in the IPv6 context. In particular, the Optimistic DAD approach presented in [53], allows under certain assumptions, the use of a particular IP address that has not yet successfully completed the DAD process. Therefore, this work is the first effort to improve the DAD procedure at DHCP servers in IPv4.

## 4.2 Standard DHCP procedure

First, we review the standard DHCP [12] procedure and identify why the current approach suffers from long delay. Fig. 4.2 shows the DHCP procedure. First, clients broadcast a DHCP DISCOVER packet to request a new IP address from the local DHCP server. When the DHCP server receives the DISCOVER packet, it chooses an unused IP address from its IP address pool and performs DAD. As a DAD procedure, the DHCP server sends ICMP echo requests to the IP address and waits for a response. When the timer expires and it has not received any ICMP response from the IP address, the IP address is assigned to the requesting client by sending a DHCP OFFER packet. When the client receives the DHCP OFFER packet, it requests the IP address from the DHCP server by sending a DHCP REQUEST packet to the DHCP server. The DHCP server checks that the requested IP address is valid when it receives the DHCP request, and it allows to use the IP address by sending a DHCP ACK packet to the client. When the client receives the DHCP ACK packet, it binds the IP address to its network layer and starts to update application sessions, if any. The assigned IP address is leased only for a certain lease time by the DHCP server. When the lease expires, clients need to update this lease by sending a DHCP REQUEST packet to the server.

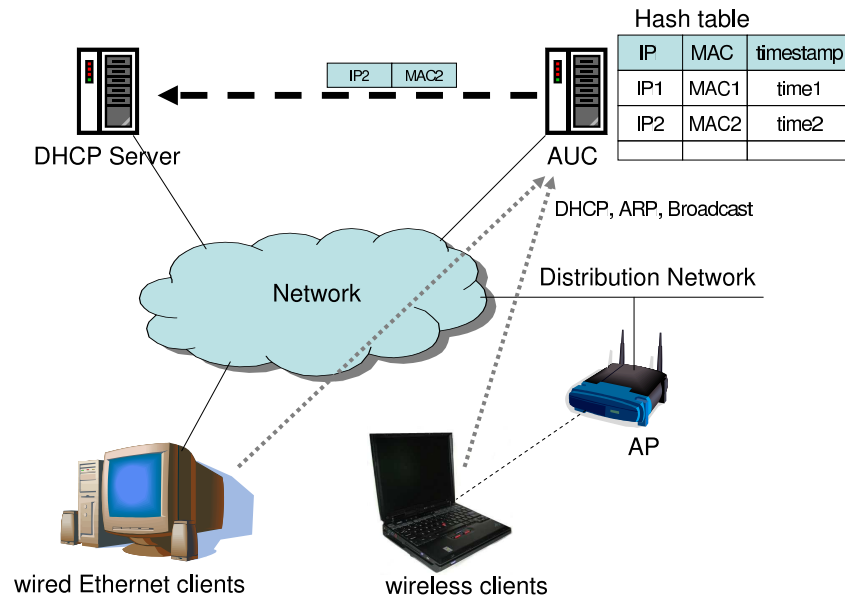


Figure 4.3: Framework of pDAD

The first problem of the DAD procedure is that host firewall software blocks the incoming ICMP echo request packet for security reasons, and thus, DAD using ICMP ECHO is not safe. The more critical problem for real time services is that it takes a long time, and the network connectivity disruption is not acceptable for such a service. We will see how pDAD can solve the problems in the following sections.

### 4.3 Framework of pDAD

Fig. 4.3 shows the framework of the pDAD, and as we can see in the figure, pDAD adds a new network component called Address Usage Collector (AUC), which interacts with the DHCP server.

#### 4.3.1 AUC

The Address Usage Collector (AUC) collects information on IP address usage by monitoring ARP and broadcast traffic for a particular subnet. To monitor such traffic in an efficient manner, the AUC should be installed on a network component that is traversed by most of the network traffic such as a router. Usually, the AUC is installed in the DHCP Relay Agent (RA) which is installed on a router of a particular subnet.

By monitoring ARP and broadcast traffic, the AUC builds a table where each entry contains an IP address, the corresponding MAC address, and a timestamp of the entry creation time. Every time a new entry is added to the table, the AUC sends a packet to the DHCP server that includes the IP address and MAC address pair. This information tells the DHCP server that a node with the MAC address is using the IP address, and therefore the IP address should not be assigned to anyone else. Figs. 4.4 and 4.5 show

IP Address	MAC Address	Timestamp
------------	-------------	-----------

Figure 4.4: Structure of entries in the AUC's table

Subnet Identifier (4 B)
MAC Address (6 B)
IP Address (4 B)

Figure 4.5: Structure of packets sent by the AUC to the DHCP server

the structure of an entry in the AUC's table and the structure of the packet sent by the AUC to the DHCP server.

To keep the information about IP addresses currently in use up to date, the AUC removes an entry from the table when its timer for that entry has expired. If the IP address for that entry is still in use after the removal of the entry for the IP address, a new entry for this IP address will be added to the table when the AUC detects the IP address.

### 4.3.2 DHCP server behavior

When the DHCP server receives a packet from the AUC, it checks if such an address was legally assigned or not. If the IP address is in the unassigned IP pool, it means that such address was illegally taken, the DHCP server then removes it from the unassigned IP address pool, and registers it to a bad-IP address list which will also mark the IP address as currently in use. In the bad-IP address list, there is a similar mechanism to the one used in the AUC's table where each entry has a timestamp. An IP address in the bad-IP address list is removed from the list when its timer has expired. This way, the DHCP server has always up-to-date information on IP addresses currently in use.

By using pDAD, the DHCP server has also much more control over the network. For example, the DHCP server could configure the packet flow rules in the egress router that block the IP addresses that have been illegally acquired by malicious users. Furthermore, some form of intrusion detection could be also implemented.

In addition to the previous considerations, pDAD also allows the DHCP server to know about duplicate addresses as they occur and not just when a client requests an IP address. In such a scenario, the DHCP server forces the legitimate user to renew the IP address, by using the DHCP FORCERENEW message [78]. Such an action cannot be forced to the malicious user as the malicious user does not use the DHCP infrastructure.

## 4.4 Experiments

To verify the performance of pDAD, it was implemented using the ISC DHCP software package (dhcpcd) [28], which is probably the most widely used DHCP server today. It was modified to handle packets from the AUC, and the AUC functionality was implemented into the relay agent contained in the dhcpcd software package.

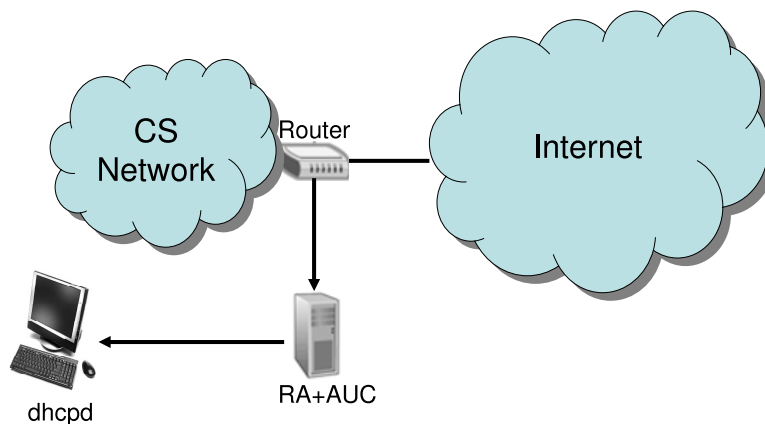


Figure 4.6: Experimental setup

#### 4.4.1 Experimental setup

The experiment was performed in the network of the Department of Computer Science, Columbia University, as shown in Fig. 4.6.

Dhcpd was installed on a desktop machine equipped with a 3 GHz Pentium 4 processor and 1GB RAM, and the RA+AUC was installed on a linux server with a 3 GHz Pentium 4 processor and 1 GB RAM. Linux kernel 2.6 was used on both machines.

The dhcpd processed packets from the AUC only, and the RA worked only as an AUC. No DHCP traffic was generated in the infrastructure itself, and the original DHCP server for the CS network was assigning IP addresses for the network. This was done in order to measure traffic and CPU load caused by pDAD only. The router of CS network forwarded all incoming and outgoing packets of the CS network to the server in which the AUC was installed. In order to collect IP address and MAC address information, the AUC module in the RA sniffed all broadcast and ARP packets from the router of the CS network. The AUC then transmitted the address information packets to the DHCP server via Ethernet.

#### 4.4.2 Experimental results

The experiments were performed a few times throughout a period of two weeks, and the performance and overhead of pDAD were measured.

##### IP address usage collection

Fig. 4.7 shows the distribution of the number of new IP addresses the DHCP server detected in an experiment. It detected 2092 IP addresses during one day, and around 1800 IP addresses among 2092 IP addresses, about 86%, were detected within an hour and a half. 47 IP addresses were detected per second at peak periods.

In order to verify the measurement results, the IP addresses that AUC has detected were compared with the DHCP log acquired from the administrator of the CS network, and it was confirmed that AUC had indeed detected all IP addresses assigned by the infrastructure DHCP server during the day.

As shown in Table 4.1, some MAC addresses had multiple IP address mappings; for example, 77 IP addresses were mapped to a single MAC address. The reason was that a firewall with proxy ARP

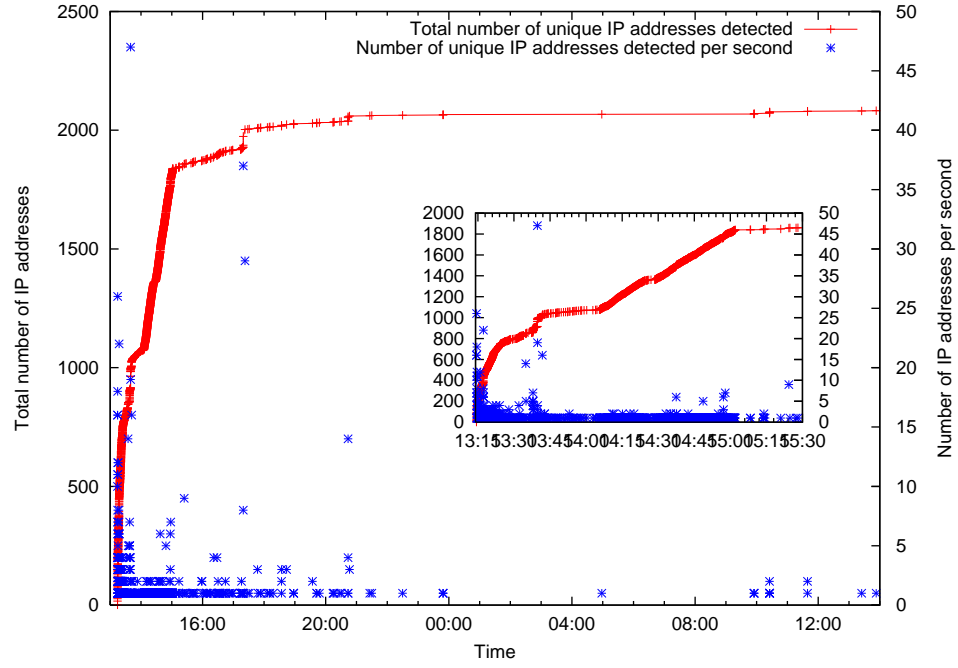


Figure 4.7: Number of new IP addresses detected by DHCP

Table 4.1: Observed number of MAC addresses with multiple IP addresses

Number of IP addresses mapped to a MAC	2	3	4	6	9	10	77
Occurrences	13	3	1	3	1	1	1

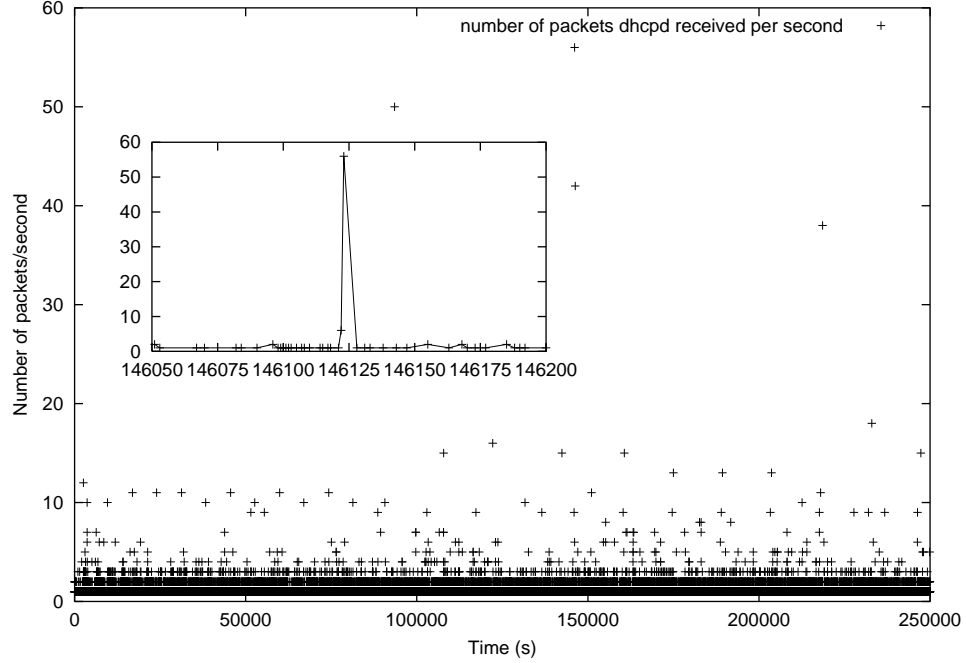


Figure 4.8: Traffic volume between DHCP server and relay agent

enabled was installed in the node, and the node was responding to all the ARP requests for nodes inside the firewall. In another case, a node requested multiple IP addresses from the DHCP server legitimately, and it was identified as a VPN server of a lab. Also, the AUC detected 136 unique IP address collisions caused by a node with MAC address 'ee:ee:80:xx:xx:xx', which appears to be a malicious node because the 'ee:ee:80' prefix is not registered as public Organizationally Unique Identifier (OUI).

#### Overhead incurred by DHCP server

Fig. 4.8 shows the traffic load between AUC and DHCP server during the experiment. The inset graph shows the same result of peak time, where the AUC sent 56 packets per a second. However, only one pair of IP address and MAC address was a new entry among them, and the rest of them were already in the table of DHCP server, which means that 55 IP addresses whose entries expired within a second coincidentally were detected.

Fig. 4.9 shows the cumulative distribution function of the number of packets per second the DHCP server received from the AUC. We can see that the DHCP server received fewer than 10 packets per second from the AUC 99% of the time, and thus the network overhead is also very small. Each packet sent by the AUC to the DHCP server contains a pair of IP address and MAC address and the RA IP address. The packet payload is 14 bytes as shown in Fig 4.5 in Section 4.3.1, bringing the total (payload + headers) packet size to 80 bytes. So, the bandwidth at peak time is 4480 B/s ( $56 \text{ packets} \times 80\text{B}$ ), and usually less than 800 B/s.

Also, because of the small amount of the traffic to the DHCP server, the additional CPU load of the DHCP server to process those packets is negligible, as was confirmed from the experiments.

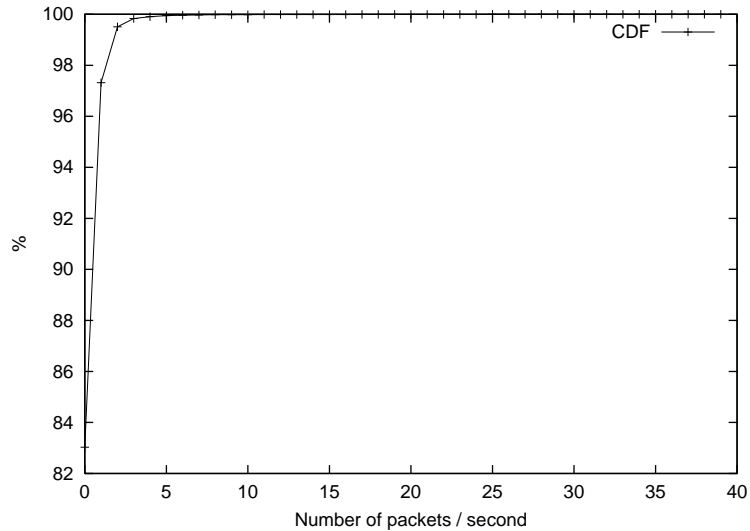


Figure 4.9: Cumulative distribution function of number of packets per second DHCP server received

### Overhead of AUC

In the experiment, AUC received 10,200 packets every second on average (Fig. 4.12). However, the AUC has processed only 1% of the packets because it needs to process only APR and broadcast packets to collect IP address usage and discards other packets; the AUC processed less than 100 packets per second 90% of the time and 273 packets per second at peak periods (Fig. 4.10 and 4.11).

Figs. 4.12 and 4.13 show the CPU load of the AUC and the correlation with the number of all packets AUC received every second. The AUC used around 40% of the total CPU power at peak time, but for 90% of the time, the CPU load was less than 20%. As Fig. 4.12 has shown, the CPU load of the AUC is exactly proportional to the traffic volume to the AUC, which means that CPU was used mostly in filtering uninteresting packets such as unicast and ARP packets from the router. It can be also inferred from that only less than 1% of the packets AUC received were used by AUC to collect IP address usage. This is because AUC received both incoming and outgoing packets of the CS network, even though AUC needs to monitor only the outgoing packets from the CS network. Therefore, the CPU load can be significantly reduced if AUC can receive only outgoing packets from the CS network.

Additional experiments were performed in the Columbia University wireless network to measure the performance in a wireless network, where clients join and leave the network more frequently, and pDAD showed similar performance and overhead with those in the wired CS network.

## 4.5 Conclusion

In this chapter, I explained a new DAD mechanism called Passive DAD, which does not introduce any overhead or additional delay during the IP address acquisition time, by introducing a new component, the AUC. AUC collects IP address usage information by monitoring the entire traffic in the subnet and updates in real time the IP address pool and a bad-IP address list in the DHCP server. Thus, when a client requests a new IP address, the DHCP server can assign an unused IP address without additional DAD procedure.



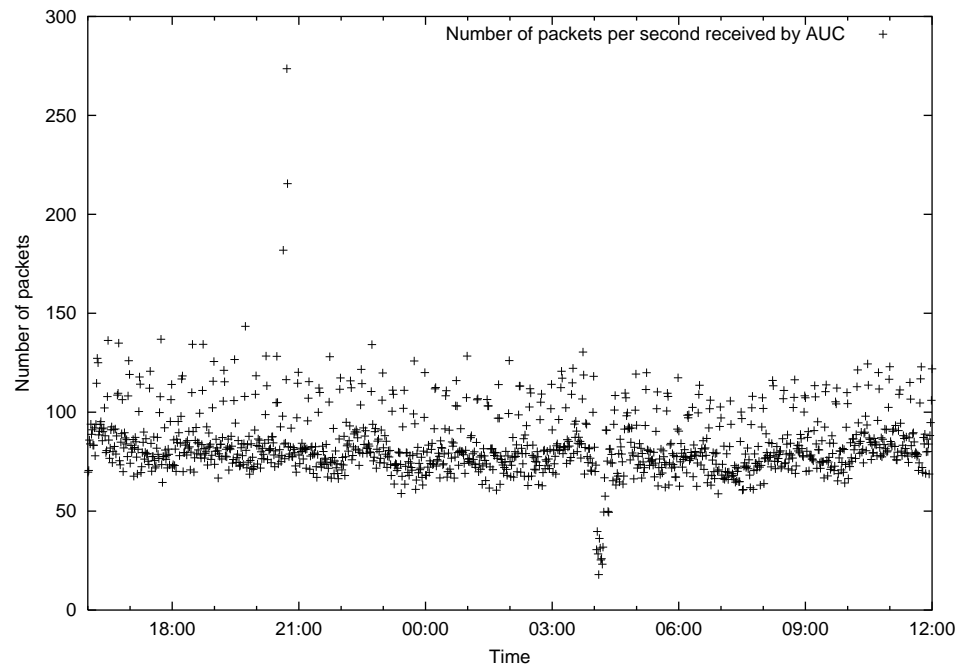


Figure 4.10: ARP and broadcast traffic volume to the AUC

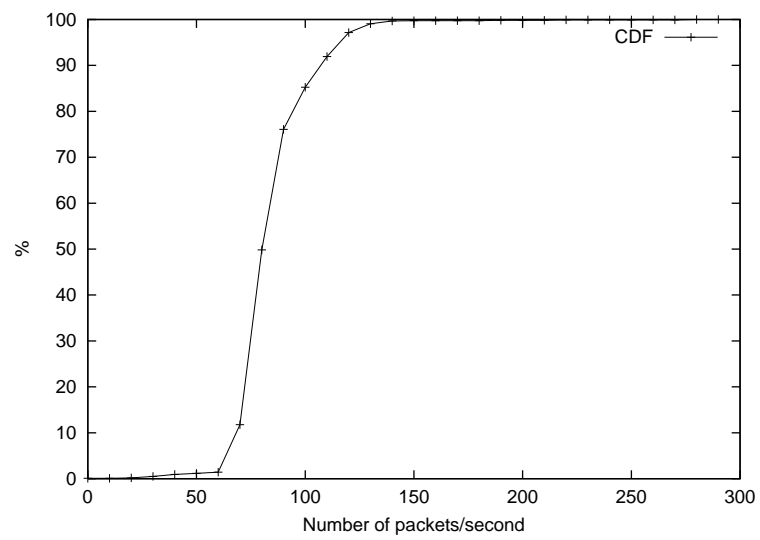


Figure 4.11: CDF of the ARP and broadcast traffic to the AUC

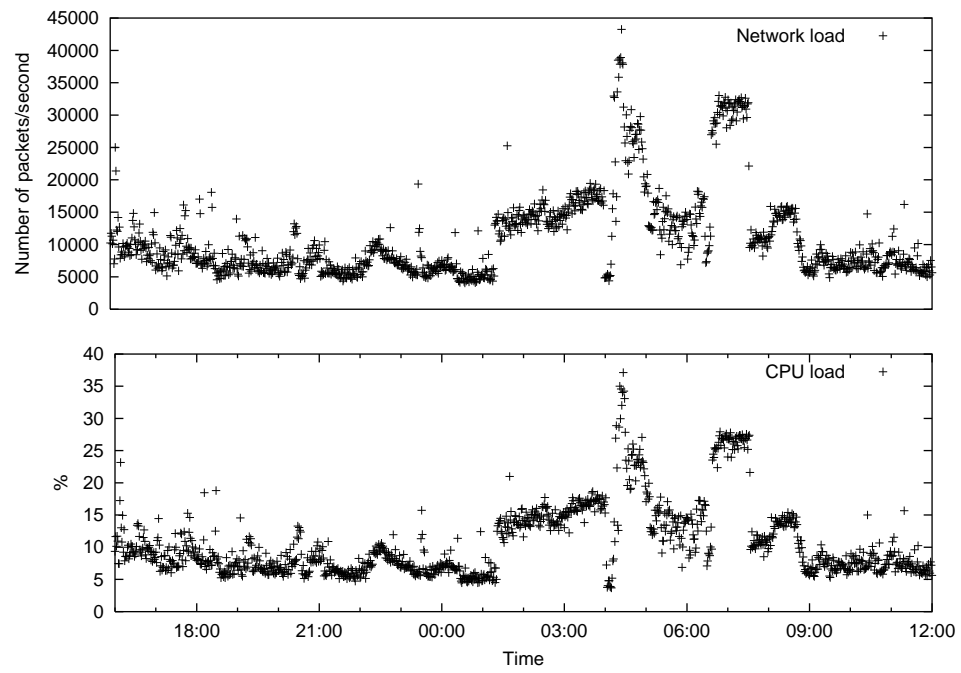


Figure 4.12: Timeline of CPU load of AUC and traffic volume received by AUC

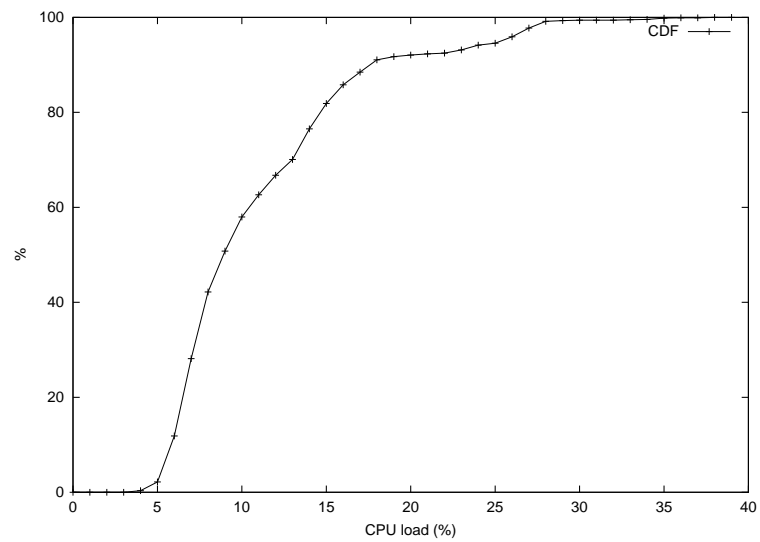


Figure 4.13: Cumulative distribution function of CPU load of AUC

Therefore, pDAD is particularly efficient in mobile environments where handoff delays can be critical for real-time communication. We can easily estimate the layer 3 handoff time from Fig. 3.9 using pDAD in the visited network, by replacing the 138 ms IP acquisition time with 20 ms, which was the average round trip time of a DHCP packet in the experiments, and the total layer 3 handoff times including all the session update become 79 ms and 57 ms with no lease and expired lease, respectively, which allow seamless handoffs for clients.

Also, pDAD performs DAD more accurately than the current DAD method does using ICMP ECHO because incoming ICMP ECHO request packet can be blocked in many host firewalls. Additionally, it helps identifying malicious users by detecting illegal IP address use in real time.

## **Part II**

# **QoS and VoIP Capacity**

# Chapter 5

## The VoIP Capacity of IEEE 802.11 WLANs

### 5.1 Introduction

Most papers have used simulations to measure the performance of new or current MAC (Media Access Control) protocols in IEEE 802.11 wireless networks because of the difficulty of implementing the MAC protocols, which are contained in the firmware of wireless interface cards. In particular, most of the papers about the capacity for VoIP traffic, including [22], [83] and [9], have used simulation tools to measure the capacity due to the necessity of a large number of wireless clients and the difficulty of controlling them and collecting data. To the best of my knowledge, very few studies, such as [18] and [41]), have measured the VoIP capacity experimentally in IEEE 802.11 wireless networks, however, without any comparison with simulation results. Also, many of them failed to take into account important parameters that affect the capacity, which resulted in each paper reporting different capacities in each paper.

In this chapter, the VoIP capacity of IEEE 802.11 wireless networks is measured using actual wireless clients in a test-bed, and the results are compared with the theoretical capacity and our simulation results. Additionally, factors that can affect the capacity but are commonly overlooked in simulations are identified, and the effect of these factors on the capacity is analyzed in detail.

### 5.2 Theoretical capacity for VoIP traffic

First, we analyze the capacity for VoIP traffic theoretically to get an upper bound, and compare it with the capacity observed in simulations and experiments.

#### 5.2.1 Capacity for CBR VoIP traffic

This section analyzes the capacity of Constant Bit Rate (CBR) VoIP traffic numerically. The theoretical capacity for VoIP traffic is defined as the maximum number of calls that are allowed simultaneously for a certain channel bit rate [35], and it is assumed that all voice communications are full duplex.

A CBR VoIP client generates one VoIP packet every packetization interval, and the packet needs to be transmitted within the packetization interval to avoid the accumulation of delay. Thus, the number

of VoIP packets that can be transmitted during one packetization interval is the maximum number of simultaneous calls, and it is the capacity for the CBR VoIP traffic. Therefore, we can compute the capacity for VoIP traffic as follows:

$$N_{CBR} = P/(2 \cdot T_t), \quad (5.1)$$

where  $N_{CBR}$  is the maximum number of CBR calls,  $P$  is the packetization interval, and  $T_t$  is the total transmission time of one voice packet including all the overhead.  $T_t$  is multiplied by 2 because the voice communication is full duplex.

The transmission of a VoIP packet entails MAC layer overhead, namely, DIFS, SIFS, ACK, and backoff time. To get an upper bound, transmissions are assumed not to incur collisions. Thus,  $T_t$  can be calculated as follows:

$$T_t = T_{DIFS} + T_{SIFS} + T_v + T_{ACK} + T_b, \quad (5.2)$$

where  $T_v$  and  $T_{ACK}$  are the time for sending a voice packet including all headers and an 802.11 ACK frame, respectively,  $T_b$  is the backoff time,  $T_{DIFS}$  and  $T_{SIFS}$  are the lengths of DIFS and SIFS. The backoff time is the number of backoff slots  $\times T_s$ , where  $T_s$  is a slot time, and the number of backoff slots has a uniform distribution over  $(0, CW_{min})$  with an average of  $CW_{min}/2$ .

Many papers, including [22] and [85], use Eqs. 5.1 and 5.2 to compute the theoretical VoIP capacity. However, many of them result in different capacity regardless of the same VoIP traffic configuration. This is because the effect of the following factors has been overlooked.

### Computation of backoff time

Backoff is performed right after a successful transmission and affects the transmission delay only when a wireless client tries to transmit frames right after the prior frame is transmitted. Therefore, the backoff does not affect the uplink delay because wireless clients transmit a packet every packetization interval, which is typically 10 ms to 40 ms, and because the uplink delay remains very low even if the number of VoIP sources reaches the channel capacity [72]. According to our experiments, the average uplink delay is less than 3 ms when the channel reaches its capacity. Thus, the backoff is added only to the downlink traffic<sup>1</sup>.

Therefore, the VoIP traffic capacity ( $N_{CBR}$ ) can be expressed as:

$$N_{CBR} = \frac{P}{2(T_{DIFS} + T_{SIFS} + T_v + T_{ACK}) + T_s \cdot CW_{min}/2} \quad (5.3)$$

Wang et. al. [85] include the backoff time in both uplink and downlink delay, resulting in a smaller capacity than the simulations and experimental results in this study. Hole and Tobagi [22] include the backoff time of the AP only, however, because they assume client backoff is done during the backoff time of the AP. Even though the assumption is acceptable, the uplink backoff time can be ignored for the reason mentioned above, regardless of the assumption.

---

<sup>1</sup>Nodes start backoff again when they sense busy medium during DIFS, but still we can ignore the backoff in uplink because we assume no collisions.

Table 5.1: Parameters in IEEE 802.11b (11 Mb/s)

Parameters	Time ( $\mu s$ )	Size (bytes)
PLCP Preamble	72.00	18
PLCP Header	24.00	6
MAC Header+CRC	24.73	34
IP+UDP+RTP headers	29.09	40
Voice	116.36	160
ACK	56	14
SIFS	10.00	
DIFS	50.00	
Slot	20.00	
$CW_{MIN}$	31 slots	

### Computation of PLCP

PLCP (Physical Layer Convergence Protocol) is composed of the PLCP preamble and the PLCP header. The standard defines short and long preambles, which are 72 bits and 144 bits, respectively, and they are transmitted with 1 Mb/s channel rate. The PLCP header size is 48 bits for both cases. However, while the PLCP header is transmitted using 1 Mb/s in the long PLCP preamble, 2 Mb/s is used in the case of the short PLCP preamble. Therefore, the PLCP transmission time is  $192 \mu s$  (PLCP preamble of  $144 \mu s$  + PLCP header of  $48 \mu s$ ) with the long preamble, and  $96 \mu s$  (PLCP preamble of  $72 \mu s$  + PLCP header of  $24 \mu s$ ) with the short preamble. In this study, the short preamble is used for comparison with the experimental results using actual wireless nodes, which also use the short preamble. Most papers use the long preamble. Only Hole et. al. [22] mention the effect of the preamble size briefly without giving analytical or simulation results. The effect of the preamble size will be discussed in Section 5.5.1.

### Transmission time of ACK frames

The rate at which ACK frames are transmitted is not clearly specified in the standard, and simulators use different rates; for IEEE 802.11b, ns-2 [48] uses 1 Mb/s by default, and the QualNet simulator [64] uses the same rate as the data packet rate. The Atheros wireless cards in the ORBIT wireless test-bed<sup>2</sup> [66] use 2 Mb/s to transmit ACK frames. Thus, 2 Mb/s is used in this study for the comparison with the experimental results. The effect of the transmission rate of ACK frames will be described in Section 5.5.4.

As the voice codec, G.711, a 64 kb/s codec and 20 ms packetization interval is used, which generate 160 byte VoIP packets not counting the IP, UDP, and RTP [70] headers. MAC layer parameters are taken directly from the IEEE 802.11b standard [23]. All the parameters used in the analysis are shown in Table 5.1.

Using the parameters mentioned above and Eq. 5.1, the theoretical capacity for 64 kb/s CBR VoIP traffic was computed to be 15 calls.

### 5.2.2 Capacity for VBR VoIP traffic

Typically, the conversations via phone calls are half duplex rather than full duplex considering that when one side talks, the other side remains silent. Thus, in order to avoid wasting resources, silence suppression can be used, which prevents sending background noise, generating VBR VoIP traffic. The VBR VoIP

<sup>2</sup><http://www.orbit-lab.org>

Table 5.2: Voice pattern parameters in ITU-T P.59

Parameter	Average duration (s)	Fraction (%)
Talkspurt	1.004	38.53
Pause	1.587	61.47
Double Talk	0.228	6.59
Mutual Silence	0.508	22.48

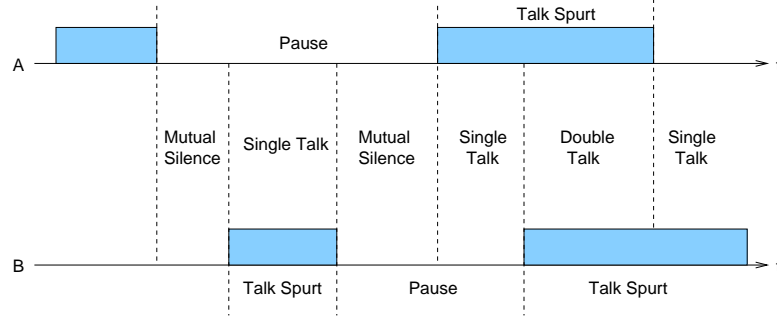


Figure 5.1: Conversational speech model in ITU-T P.59

traffic is characterized by on (talking) and off (silence) periods, which determine the activity ratio and also the capacity for VBR VoIP traffic. The activity ratio is defined as the ratio of on-periods and the whole conversation time.

In this analysis, the conversational speech model with double talk described in ITU-T P.59 [30] is used. The parameters are shown in Table 5.2, and the conversation model is shown in Fig. 5.1. The activity ratio in the conversational speech model is about 0.39 based on the fraction of talkspurts in Table 5.2.

The difference between CBR and VBR traffic is the number of packets generated every second; while a CBR VoIP source with 20 ms packetization interval generates 50 packets, a VBR VoIP source with the same packetization interval and 0.39 activity ratio generates 19.5 packets on average every second. Thus, to deduce the capacity for VBR traffic ( $N_{VBR}$ ), we rewrite Eq. 5.1 as follows:

$$N_{CBR} = \frac{1}{\frac{1}{P} \cdot 2 \cdot T_t}, \quad (5.4)$$

which means that CBR VoIP traffic generates  $1/P$  packets every second, and the capacity is computed as the number of packets that can be transmitted per time unit. In VBR,  $\alpha/P$  ( $\alpha$  is the activity ratio) packets are generated, and we deduce  $N_{VBR}$  as follows, replacing  $1/P$  with  $\alpha/P$ :

$$N_{VBR} = \frac{1}{\frac{\alpha}{P} \cdot 2 \cdot T_t}, \quad (5.5)$$

We can see that Eq. 5.5 becomes the capacity for CBR when  $\alpha$  is 1, and finally, Eq. 5.5 becomes:

$$N_{VBR} = N_{CBR}/\alpha, \quad (5.6)$$

Using Eq. 5.6, the capacity for the VBR VoIP traffic with 0.39 activity ratio is computed to be 38 calls.



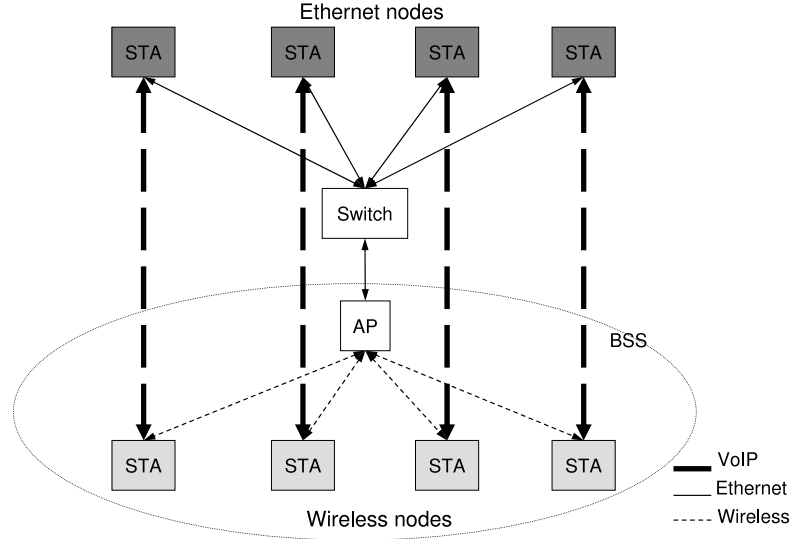


Figure 5.2: Simulation topology

### 5.3 Capacity for VoIP traffic via simulation

In this section, the capacity for VoIP traffic is measured via simulations using the QualNet simulator [64], which is a commercial network simulation tool and known to have a more realistic physical model than other tools such as ns-2 [69] [77].

In order to determine the capacity for VoIP traffic, the 90th percentile<sup>3</sup> delay at each node was collected with a varying number of wireless nodes. The one-way end-to-end delay of voice packets is supposed to be less than 150 ms [29]. The codec delay is assumed to be about 30-40 ms at both sender and receiver, and the backbone network delay to be about 20 ms. Thus, the wireless networks should contribute less than 60 ms delay [35]. Therefore, the capacity of VoIP traffic is defined as the maximum number of wireless nodes so that the 90th percentile of both uplink and downlink delay does not exceed 60 ms.

#### 5.3.1 Simulation parameters

The Ethernet-to-wireless network topology (Fig. 5.2) was used for simulations to focus on the delay in a BSS. In the simulations, the Ethernet portion added 1 ms of transmission delay, which allows us to assume that the end-to-end delay is essentially the same as the wireless transmission delay. The same parameters in Table 5.1 are used in simulations. Each simulation ran for 200 seconds and was repeated 50 times using different seeds and VoIP traffic start time. (The effect of the traffic start time will be explained in Section 5.5.3.)

<sup>3</sup>To measure the QoS for VoIP traffic, 90th percentile value is used to capture the fluctuation of the end-to-end delay, which will be contributed to a fixed delay by a playout buffer.

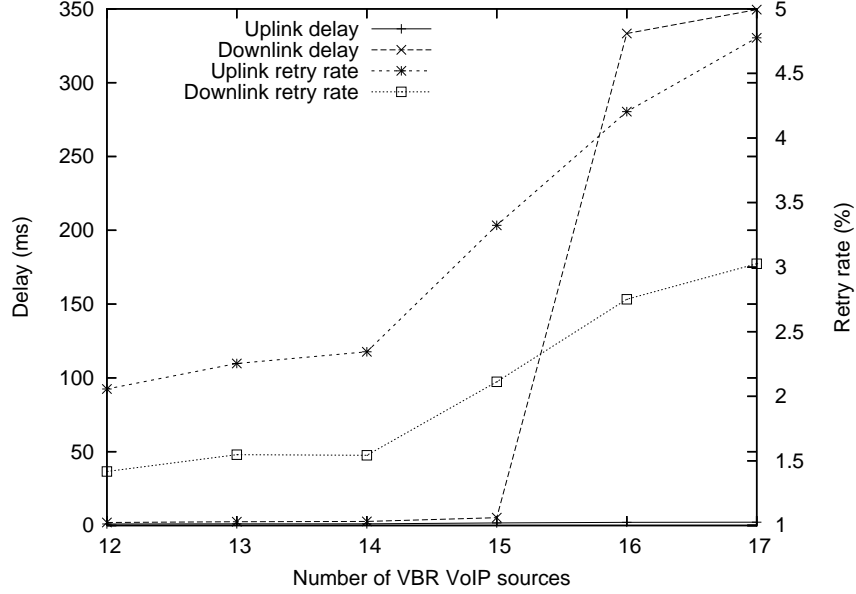


Figure 5.3: 90th percentile delay and retry rate of CBR VoIP traffic in simulations

### 5.3.2 Capacity for CBR VoIP traffic

In order to determine the capacity for VoIP traffic, the 90th percentile end-to-end delay of each VoIP flow was collected, and the average of them was calculated in each simulation and the average of all simulation results was computed. Fig. 5.3 shows the average of the 90th percentile delay of CBR VoIP traffic across simulations. The figure shows that the capacity for the VoIP traffic is 15 calls, the same as the theoretical capacity. The reason that the simulation result with collisions is the same as the theoretical capacity with no collisions is that in simulations many nodes decrease their backoff counters simultaneously while in the theoretical analysis they are counted separately. The results will be analyzed in detail later, compared with the experimental results.

### 5.3.3 Capacity for VBR VoIP traffic

The VBR VoIP traffic with 0.39 activity ratio was implemented in the QualNet simulator, with exponentially distributed on-off periods, following the speech model described in Section 5.2.2. Fig. 5.4 presents the delay and retry rate for VBR VoIP traffic. The downlink delay increases slowly compared with that of CBR VoIP traffic, and this is because only 50 kb/s ( $64 \text{ kb/s} \times 2 \times 0.39$ ) VoIP traffic is added to network as one VBR call is added. As we can see, the capacity of VBR VoIP traffic is 38 calls, the same as the theoretical capacity.

## 5.4 Capacity for VoIP traffic via experiments

I performed experiments to measure the capacity for VoIP traffic in the ORBIT (Open Access Research Testbed for Next-Generation Wireless Networks) test-bed, which is a laboratory-based wireless network emulator located at WINLAB, Rutgers University, NJ.

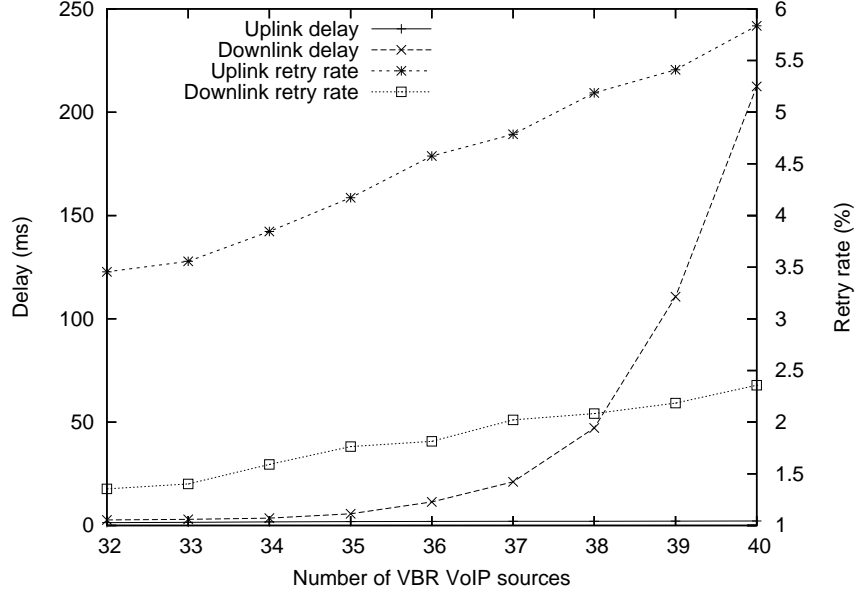


Figure 5.4: 90th percentile delay and retry rate of VBR VoIP traffic in simulations

#### 5.4.1 The ORBIT test-bed

ORBIT is a two-tier laboratory emulator and field trial network test-bed designed to evaluate protocols and applications in real-world settings [66]. The ORBIT test-bed is composed of a main grid called 'grid' with  $20 \times 20$  nodes and multiple smaller test-beds.

The main grid was used, which consists of 380 nodes with Atheros chipset (AR5212) wireless cards (Atheros nodes) and 20 nodes with Intel chipset wireless cards, and it forms a  $20 \times 20$  grid with one meter inter-node distance. Every node has a Pentium IV CPU with 1GB memory, runs Linux (kernel 2.6.19). It has two wireless and two Ethernet interfaces, and the MadWifi driver 0.9.2 is used as the wireless card driver. A center node was set up as the AP so that distances between the AP and nodes are within 10 meters, which is close enough to avoid the effect of signal strength on packet loss. The RSSI (Received Signal Strength Index) of each node was also analyzed and will be explained later.

A simple UDP client was used to send 172 byte (160 B VoIP payload + 12 B RTP header) UDP packets to a specified destination. The UDP client records the sending time and receiving time in separate files with the UDP sequence number, which is included as the UDP packet payload, the data were used to calculate the downlink and uplink delay and the packet loss. In order to synchronize the system clock of the nodes, the Network Time Protocol (NTP) [50] was used. Every node updated the system clock every several seconds using the *ntpdate* application through Ethernet because the system clock of each node started to skew slightly a few seconds after updating the clock.

The MadWifi driver was modified to print out the information of all transmitted and received frames such as RSSI, retries, and 802.11 flags, which are reported from the firmware to the driver. The information was used to calculate the retry rate and to analyze the effect of RSSI and control frames.

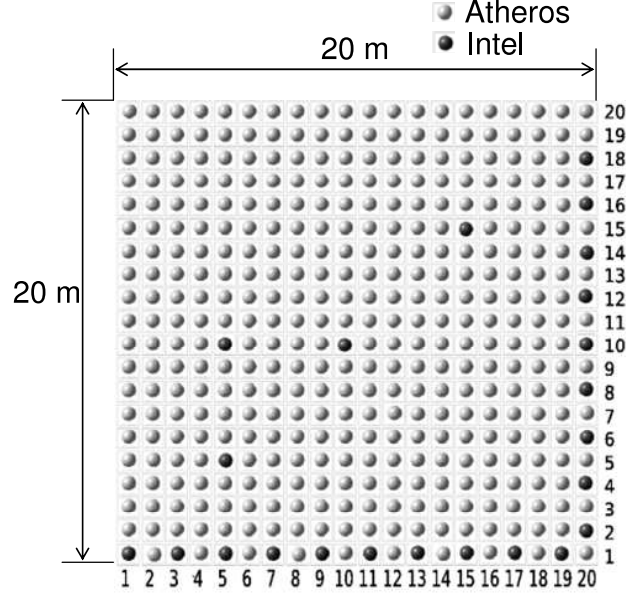


Figure 5.5: Node layout in the grid ORBIT test-bed

### 5.4.2 Experimental results

The experimental results showed higher fluctuation than the simulations across experiments. Therefore, each experiment was performed more than 10 times with a 200 s experiment time over four months.

Figs. 5.6 and 5.7 show the average 90th percentile delay and retry rate of uplink and downlink with CBR and VBR VoIP sources, respectively. We can see that the capacity of CBR VoIP traffic is 15 calls, and the capacity for VBR VoIP traffic with 0.39 activity ratio is 38 calls, which are the same as the theoretical capacity and the simulation result.

### 5.4.3 Analysis of the results and comparison with simulation results

#### Delay

As the number of VoIP sources increases, the downlink delay increase is much larger than the uplink delay increase because of the unfair resource distribution between uplink and downlink [72], as will be shown in Chapter 7. We can see this behavior in both simulations and experiments. However, the delay increase shows minor differences between the results from simulations and experiments results even though the simulations and experiments have shown the same VoIP capacity. While the uplink delay increases to 300 ms in simulations when the number of VoIP sources exceeds the capacity, it increases only to 80 ms in the experiments. This is because of the difference in the buffer size of the AP. The simulator has a 50 KB buffer and the MadWifi driver (0.9.2) limits the number of packets in the queue as 50 by default<sup>4</sup>. The bigger buffer stores more packets and increases the queuing delay. The effect of the buffer size will be discussed in detail in Section 5.5.8. The downlink delay increase is also slightly different; while the downlink delay increases almost linearly until 15 calls in the experiments, it remains very low in

<sup>4</sup>The buffer size differs according to the version of MadWifi drivers.

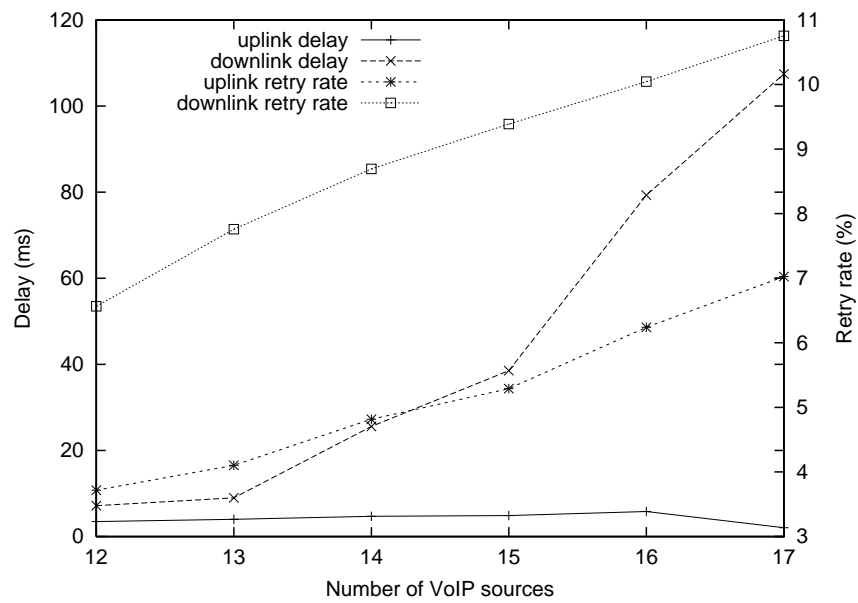


Figure 5.6: 90th percentile delay and retry rate of CBR VoIP traffic in the experiments

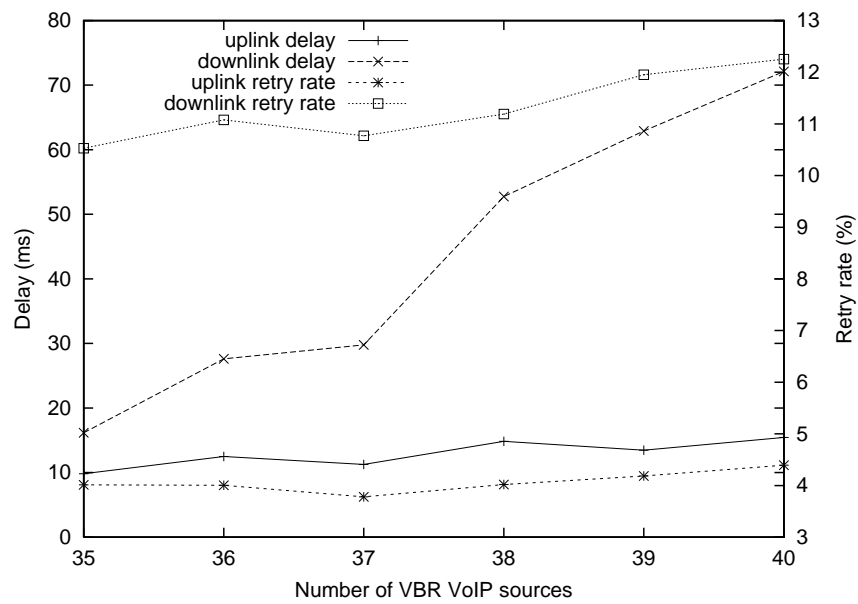


Figure 5.7: 90th percentile delay and retry rate of VBR VoIP traffic in the experiments

simulations. The reason is that the retry rate in the experiments is higher than that of the simulations. Also, we can see that the downlink delay increases slowly starting with 16 VoIP sources in both simulations and experiments. This is because of the introduction of packet loss due to the buffer overflow at the AP, and the queuing delay at the AP does not increase much even with 16 calls. The packet loss rate with 15 calls in the experiments is only 0.6% but increases to 5% with 16 VoIP sources.

### Retry rate

In both simulation and experiments, the uplink retry rate is much higher than the downlink retry rate. The reason is that uplink packets collide with packets from other clients (uplink) as well as the AP (downlink). This can be verified numerically: when the number of collisions between uplink and downlink is  $C_1$  and the number of collisions among uplink packets is  $C_2$ , the retry rate of uplink and downlink becomes  $(C_1 + C_2)/(P + C_1 + C_2)$  and  $C_1/(C_1 + P)$ , respectively, where  $P$  is the number of packets sent in each uplink and downlink, considering uplink and downlink traffic volume is the same. We assume that the uplink retry rate is always larger than the downlink retry rate. Then, the following equation should be always satisfied:

$$\frac{C_1 + C_2}{P + C_1 + C_2} - \frac{C_1}{C_1 + P} > 0$$

Then, it becomes  $C_2 \cdot P > 0$  and is always satisfied since  $C_2, P > 0$ . Accordingly, the uplink retry rate is always higher than the downlink retry rate.

## 5.5 Factors that affect the experimental and simulation results

The initial experimental results, which are not included here, showed a big difference in the capacity from the theoretical analysis and simulations, and some parameters that are commonly ignored but affect the experimental and simulation results were identified. In this section, I discuss the factors in detail with some additional experimental and simulation results.

I will focus on CBR traffic in the analysis because I want to avoid the effect of activity ratio, which is a main factor in the experimental results with VBR VoIP traffic, and because the effect of the following factors on MAC layer would be the same for both CBR and VBR VoIP traffic.

I found that the preamble size (Section 5.5.1), rate control (Section 5.5.2), VoIP packet generation offsets among VoIP sources (Section 5.5.3), and the channel transmission rate of ACK frames (Section 5.5.4) were the main factors that affect the VoIP capacity, and that the signal strength (Section 5.5.5), scanning APs (Section 5.5.6), the retry limit (Section 5.5.7), and the network buffer size (Section 5.5.8) also affect the experimental results even though they did not change the VoIP capacity in my experiments.

### 5.5.1 Preamble size

The preamble size affects the capacity for VoIP traffic in IEEE 802.11b networks, and simulators and wireless cards use various sizes. The preamble is a pattern of bits attached at the beginning of all frames to let the receiver get ready to receive the frame, and there are two kinds of preambles, long and short. The long preamble has 144 bits and the short one has 72 bits. The long preamble allows more time for the receiver to synchronize and be prepared to receive while the transmission time becomes longer, since the preamble is transmitted at a channel rate 1 Mb/s. Both the QualNet simulator and ns-2 use the long

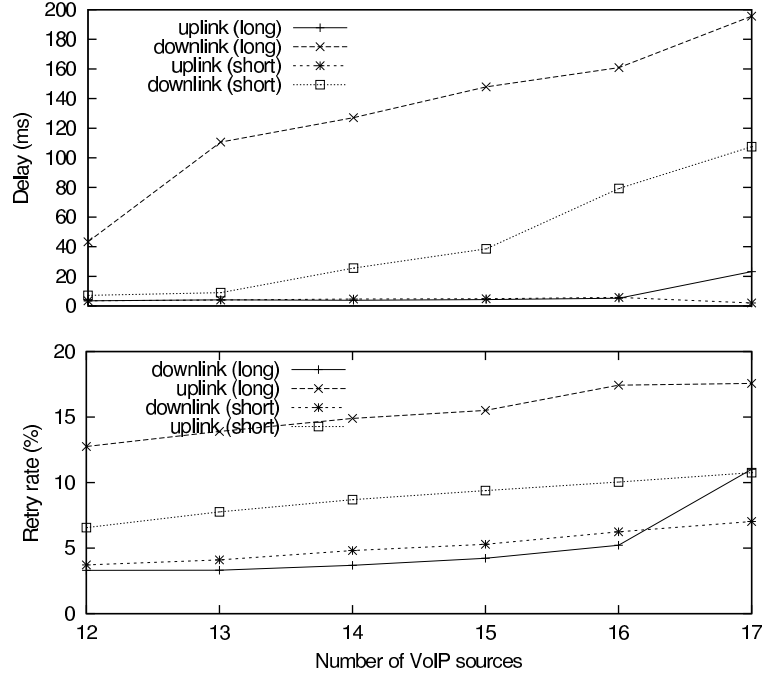


Figure 5.8: 90th percentile delay and retry rate of CBR VoIP traffic with long and short preamble via experiments

preamble by default while recent wireless cards and the drivers use the short preamble due to advancing RF technology, improving the utilization of channels. The MadWifi driver in the ORBIT test-bed uses the short preamble by default, and the type can be changed by modifying the driver source code.

Considering the small packet size of VoIP packets and the low transmission speed, the preamble takes up a big portion of a VoIP packet; 144 bits of 2064 bits, taking up 4% in size, but  $144 \mu s$  of  $362 \mu s$ , which is 40% in the transmission time. Thus, the theoretical capacity for the VoIP traffic in DCF decreases from 15 to 12 calls when the long preamble is used.

Fig. 5.8 presents 90th percentile delay and retry rate for CBR VoIP traffic using the long and short preambles in the experiments. As expected, the uplink retry rate doubled when the long preamble is used, reducing the capacity to 12 calls, which is same as the theoretical capacity using the long preamble.

### 5.5.2 Rate control

Most wireless cards support multi-rate data transmission, and wireless card drivers support Auto-Rate Fallback (ARF) to choose the optimal transmission rate according to the link status. Generally, the transmission rate decreases when the packet loss exceeds a certain threshold and increases after successful transmissions, but the specific behavior depends on the ARF algorithm.

A smart rate control algorithm improves the throughput and the channel utilization, however, only when the packet loss is caused by wireless link problems. ARF makes the channel utilization and throughput worse if the main reason for the packet loss are packet collisions [67]. In this case, the transmission with a low bit-rate extends the transmission time of frames and increases the delay without improving the packet loss, and the packet transmission with the highest available bit-rate achieves the best throughput.

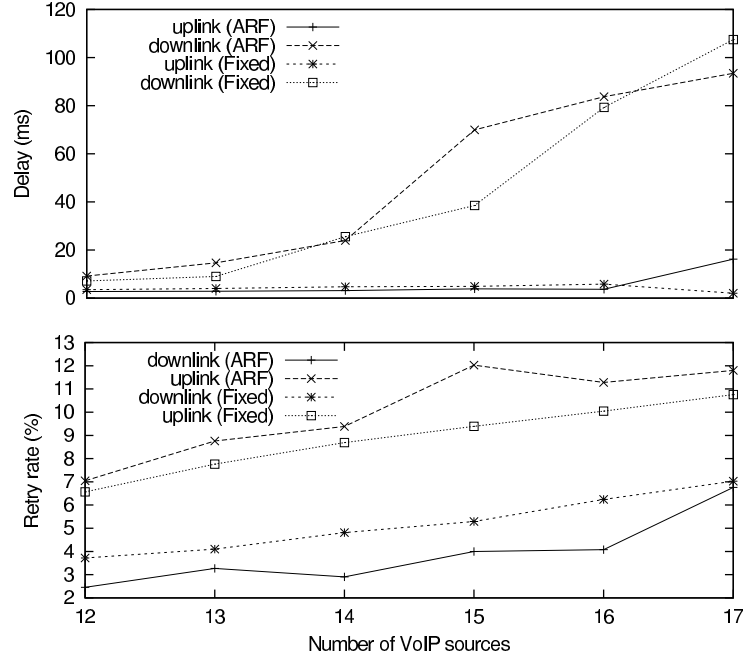


Figure 5.9: 90th percentile delay and retry rate of CBR VoIP traffic with and without the AMRR rate control algorithm in the experiments

The MadWifi driver supports three rate control algorithms, ONOE [55], Adaptive Multi-Rate Retry (AMRR) [45], and SampleRate [4]. The SampleRate control algorithm is used by default, but we can change or disable it by modifying the driver source code. Fig. 5.9 shows the experimental results with the AMRR rate control algorithm. The capacity decreases to 14 with that rate control algorithm, because with 15 CBR VoIP sources, about 8% of the packets are transmitted with lower transmission rates due to the rate control algorithm. Fig. 5.9 also shows the retry rate, and we can see that ARF helps slightly when collisions are less (downlink), but it is detrimental when more collisions happen (uplink), and it increases the delay. The effect of the rate control on the capacity depends on the algorithm and the RF conditions, and the analysis of the algorithm is beyond the scope of this study.

The QualNet simulator supports a few ARF algorithms, and ns-2 also has many external rate control modules. However, generally a fixed transmission rate is used in most simulations to avoid the effect of the rate control algorithm, while many wireless card drivers use a rate control algorithm by default. Therefore, when comparing the results from simulations and experiments, the rate control should be disabled or exactly the same rate control algorithm should be used in both simulators and the drivers of all wireless nodes.

### 5.5.3 VoIP packet generation offsets among VoIP sources

In simulations, normally all wireless clients start to generate VoIP traffic at the same time, but the packet generation offset between clients affects the simulation results.

As soon as a VoIP packet is generated at the application layer and sent to the empty network queue at the MAC layer, it is transmitted to the medium without further backoff if the medium is idle. This is



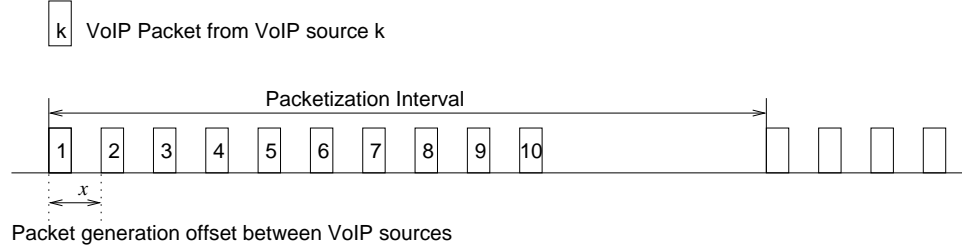


Figure 5.10: An example of VoIP packet transmission in the application layer with 10 VoIP sources with the fixed transmission offset of  $x$

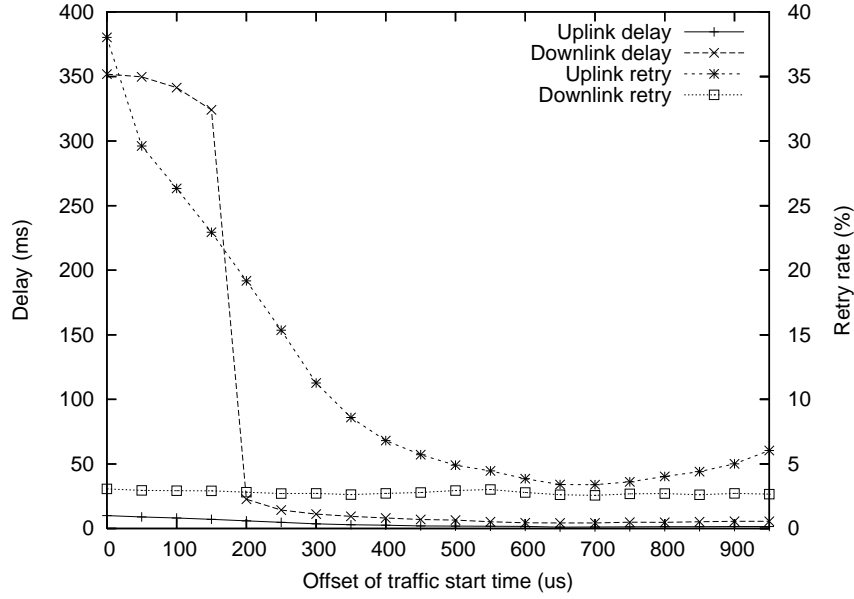


Figure 5.11: 90th percentile delay and retry rate as a function of packet generation offset among VoIP sources

because backoff is done immediately after a successful transmission, and wireless clients generate VoIP packets every packetization interval, which is typically 10 ms to 40 ms. We have shown that when the number of VoIP sources does not exceed the capacity, uplink delay is very small, which means that the outgoing queue of VoIP wireless clients is mostly empty.

Therefore, generally, when two VoIP sources generate VoIP packets at the same time, the collision probability of the two packets becomes very high. Conversely, when the VoIP packet generation times of all VoIP sources are evenly distributed within a packetization interval, the collision probability between nodes becomes lowest.

Fig. 5.11 shows the 90th percentile end-to-end delay and retry rate of the VoIP traffic in simulations with 15 VoIP sources and 0 to 950  $\mu s$  packet generation offsets. We can see that the delay decreases as the offset increases. With 200  $\mu s$  offset, the delay drops below 50 ms, changing the capacity from 14 calls to 15 calls. The retry rate becomes lowest at 650  $\mu s$  packet generation offset, and it starts to increase again after the point. This is because  $CW_{min}$  is 31 and the initial backoff time is chosen between 0 and

620  $\mu s$  in 802.11b; when the offset of two packets from two different clients is larger than 620  $\mu s$ , the two packets cannot be transmitted at the same time regardless of their backoff time, and the probability that the two packets collide each other drops to zero. However, even in this case, still collisions happen between uplink and downlink, and if the uplink packets are retransmitted due to the collisions, uplink packets can collide with other uplink packets regardless of the large offset.

We have seen that the capacity of VoIP traffic varies from 14 to 15 calls according to the offset. Therefore, in the simulations, the starting time of each VoIP source was chosen randomly between 0 to 20 ms (packetization interval), as this corresponds to the experiments.

#### 5.5.4 Channel transmission rate of Acknowledgment (ACK) frames

An ACK frame should be sent to the sender for each successfully transmitted unicast data frame. Thus, it takes a significant amount of bandwidth, and the transmission rate affects the capacity. The channel transmission rates of ACK frames are not specified in the standard, and simulators and wireless cards use different transmission rates: the QualNet simulator uses the data rate, ns-2 uses the lowest rate, and the Atheros nodes in the test-bed use 2 Mb/s by default, which can be changed by modifying the driver. Transmitting ACK frames with a lower data rate reduces the number of retransmissions due to ACK frame loss when the wireless link is unreliable, but when channels become congested, it increases collisions instead and increases the delay due to the long transmission time. Fig. 5.12 shows the retry rate and delay of VoIP traffic when ACK frames are transmitted with 2 Mb/s and 11 Mb/s. We can see that the retry rate of both uplink and downlink decrease when ACK frames are transmitted using 11 Mb/s, increasing the capacity to 16 calls, also because of the short ACK transmission time.

In the theoretical analysis, when 11 Mb/s is used for ACK frames, the transmission time reduces from 152  $\mu s$  to 106  $\mu s$  by 46  $\mu s$ , about 6% of the total transmission time of a VoIP frame including backoff, increasing the capacity for the CBR VoIP traffic from 15 calls to 16 calls, as in the experimental result.

#### 5.5.5 Received Signal Strength Indication (RSSI)

The weak signal strength can be the main reason for the frame loss in experiments, but it is commonly ignored in simulations.

Fig. 5.13 shows the RSSI of uplink and downlink flow in each node. The vertical bar shows the range of the RSSIs of a node, and the height of the box and the center line represent the standard deviation and average of RSSIs across the experiments, respectively. Moreover, in order to identify the correlation between the RSSI and the distance from the AP, the RSSI of each node is plotted according to the distance between the node and the AP.

We can see that the RSSIs of most of the nodes fall within the interval -55 to -70 dBm, and the fluctuation of RSSI of each node across the experiments was mostly within 5 dBm. Only Node 6 had a relatively weak signal, but it was still within effective range. Also, any correlation between the RSSI and the distance from the AP was not found: the weak signal was not because of the distance from the AP.

Furthermore, in order to check the effect of signal strength on the experiments, the correlation between the signal strength and the retry rate was analyzed, but no correlation was found in both downlink and uplink. This means that the signal was strong enough and any bit error was not caused directly by the weak signal. However, the frames with the RSSI below -76 dBm (from Node 6 in Fig. 5.13) had a higher retry rate than the other nodes in uplink, in particular, with 17 VoIP sources. The behavior was caused by the capture effect where the frame with the stronger signal can be captured by the receiver through a

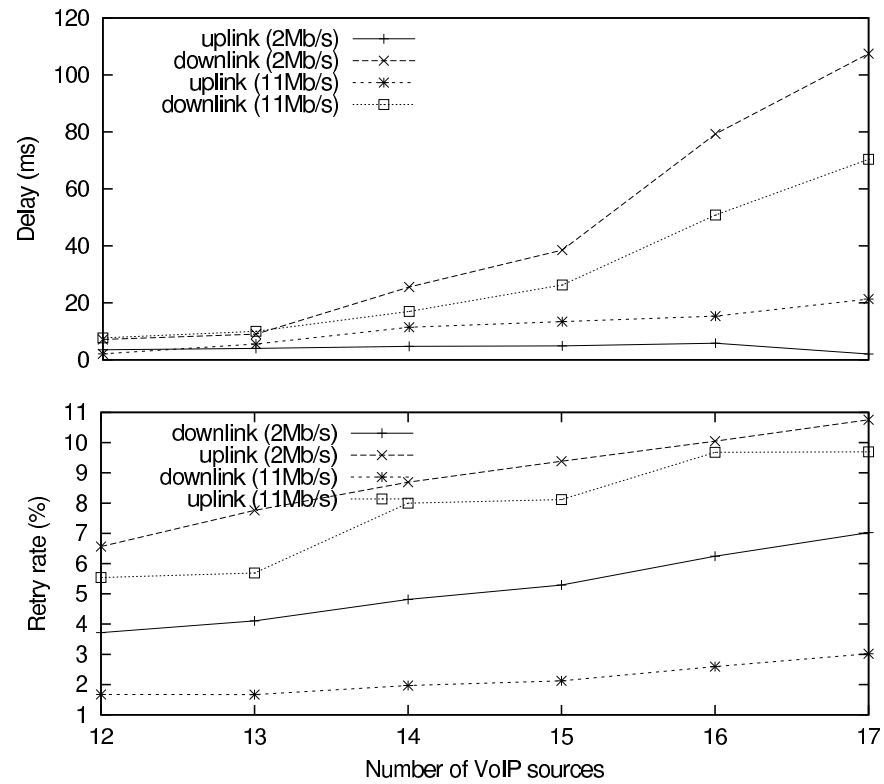


Figure 5.12: Experimental results with different ACK transmission rates

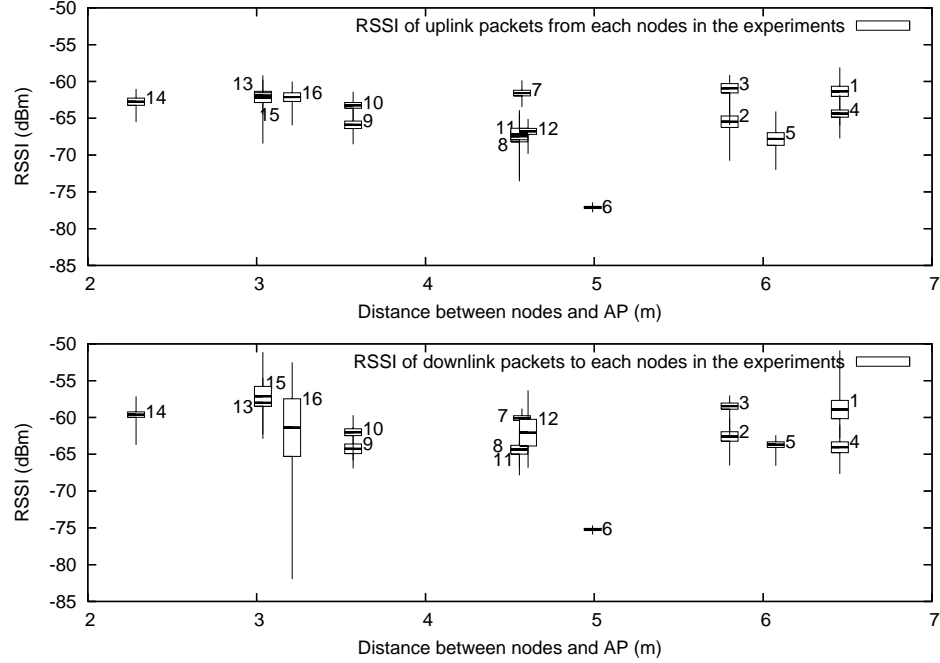


Figure 5.13: RSSI values as a function of the distances between nodes and the AP

collision [43].

### 5.5.6 Scanning APs

Probe request and response frames were observed in the experiments, while they are not observed in simulations.

Probe request frames are transmitted from wireless clients to scan APs for handoff, and response frames are transmitted from the AP [23]. Typically, wireless clients transmit a probe request frame to a channel and scan all channels, for example, 11 channels in IEEE 802.11b. The handoff procedure is implemented in the firmware of wireless cards, and each vendor uses different algorithms, as explained in Chapter 2. Thus, it is hard to determine the exact reason for transmitting probe request frames. However, typically, wireless clients start to scan APs to find better APs when the signal strength of frames from the current AP drops below a threshold and also when they experience a certain amount of packet loss [84]. It was also found in the experiments that as the number of VoIP sources increases, the retry rate and the number of probe request frames also increases (Fig. 5.14).

Probe request and response frames increase the delay of VoIP packets due to traffic increase and the higher priority of management frames over data frames. Even though the effect on the capacity of VoIP traffic was negligible in the experiments, the effect depends on the handoff algorithms of the wireless cards; for example, some wireless cards regularly scan APs for efficient handoff [84]. In this case, the effect of scanning APs becomes bigger. Therefore, the handoff behavior of wireless cards needs to be investigated before performing experiments with them.

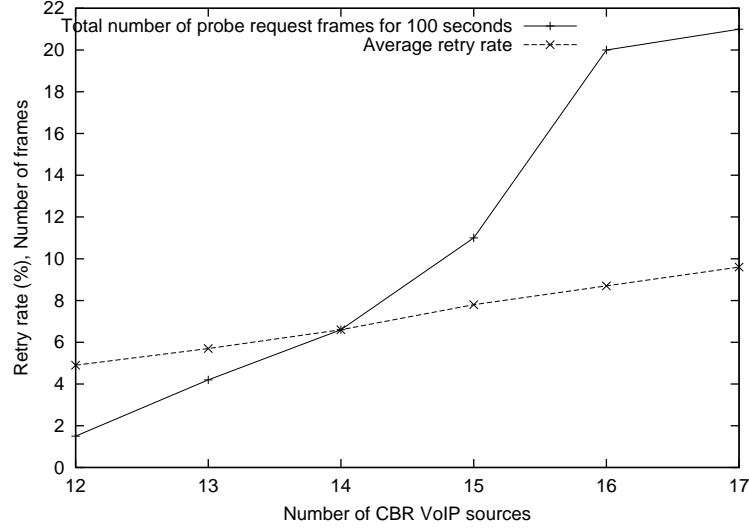


Figure 5.14: Probe request frames and retry rate in the experiments

### 5.5.7 Retry limit

IEEE 802.11 standard defines two kinds of retry limit, long and short. Unicast frames can be retransmitted until the number of retries reaches the retry limit. The short retry limit is typically used when the MAC frame size of packets is equal to or smaller than the RTS threshold value, and the long retry limit is used otherwise [23]. Although the specific values are not defined in the standard, seven and four are accepted as appropriate long and short retry limit values, and generally they are not configurable in wireless cards.

The wireless cards with the Atheros chip-set in the ORBIT test-bed used the long retry limit even if the RTS threshold value was set to off (infinite). Fig. 5.15 shows the distribution of the number of retransmissions in the experiments. According to the figure, when the number of VoIP sources exceeds the capacity, packets are retransmitted at most 11 times, which indicates that the long retry limit is 11 in the Atheros nodes. However, the QualNet simulator uses 7 and 4 as the short and long retry limit, respectively, and the short retry limit is used when the packet size is smaller than or equal to the RTS threshold value, as in the standard.

The retry limit affects the packet loss and delay. Fig. 5.15 shows that the retry limit did not cause packet loss as long as the number of VoIP sources remained below the capacity, 15 calls; there was no packet loss due to the retry limit before the number of VoIP sources reached the capacity, and the packet loss due to the retry limit is also negligible, even with 17 VoIP calls. Fig. 5.15 also shows the cumulative distribution function of the number of retransmitted packets. According to the figure, we can see that the packet loss would be the same even if 7 was used as the retry limit in the experiments, which shows that the difference in the retry limits did not affect the experiments.

### 5.5.8 Network buffer size

The packet loss rate is another metric to measure the capacity and QoS for VoIP traffic, and it is known that 1 to 3% packet loss is tolerable for VoIP traffic [29]. With 15 VoIP sources, which is the capacity of CBR VoIP traffic, the packet loss rate was only 0.6% in the experiments, which satisfies the condition.

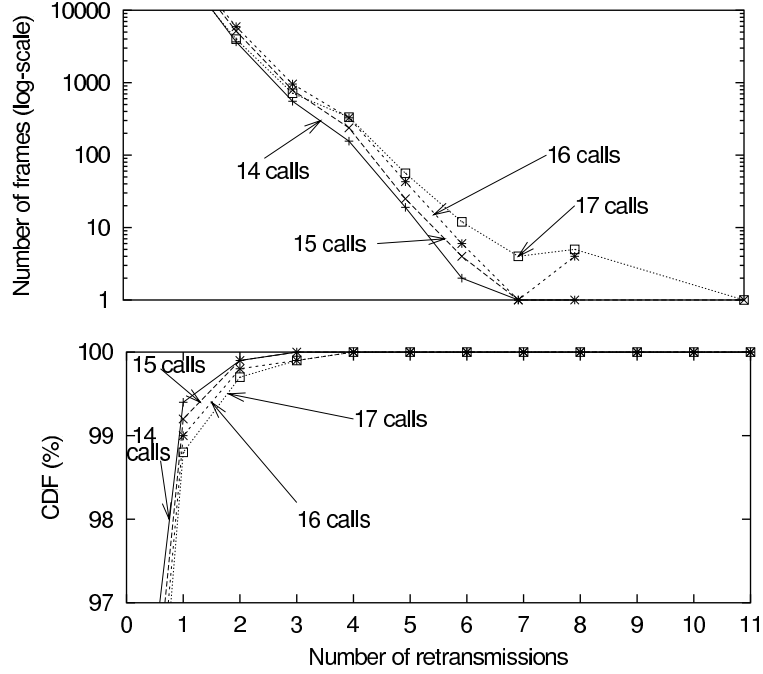


Figure 5.15: Distribution and cumulative distribution function (CDF) of number of retransmissions of CBR VoIP traffic in the experiments

The packet loss happened only in the downlink in the experiments, and we have already seen in the previous section that the packet loss due to exceeding the retry limit was almost zero, which means that it was caused by buffer overflow at the AP. Therefore, the network buffer size at the AP directly affects the packet loss as well as delay. In order to identify the relationship among the buffer size, delay, and packet loss, a formula for computing the maximum queuing delay is deduced using the buffer size. The maximum queuing delay ( $D_{max}$ ) can be easily computed as follows, using Little's Law.

$$D_{max} = M \cdot \mu_{avg} = \frac{B}{S} \cdot \mu_{avg},$$

where  $M$  is the maximum number of packets in the buffer,  $\mu_{avg}$  is the average transmission time of the IP packet,  $B$  is the buffer size, and  $S$  is the an IP packet size. For example, if a packet size is 200 B, the average transmission time of a frame at the MAC layer is 2 ms, and the buffer size of the AP is 10 KB, then 50 packets can be stored at the AP, and the maximum queuing delay is computed as 100 ms. Using this equation, we can also deduce the minimum buffer size ( $B_{min}$ ) to avoid affecting the capacity. If the threshold queuing delay for the capacity is  $D_{th}$ , then  $B_{min}$  becomes  $(S \cdot D_{th})/\mu_{avg}$ . The experiments used  $M = 50$  since MadWifi driver 0.9.2 limits the number of packets in the buffer as 50 and  $\mu_{avg} = 2$  ms with 15 calls. Then,  $D_{max}$  becomes 100 ms, which is larger than 60 ms, the threshold for the capacity. Thus, we can confirm that the buffer size of the MadWifi driver did not affect the capacity.

Therefore, we can conclude that having a longer buffer at the AP does not improve QoS of VoIP traffic, although it can reduce the packet loss.

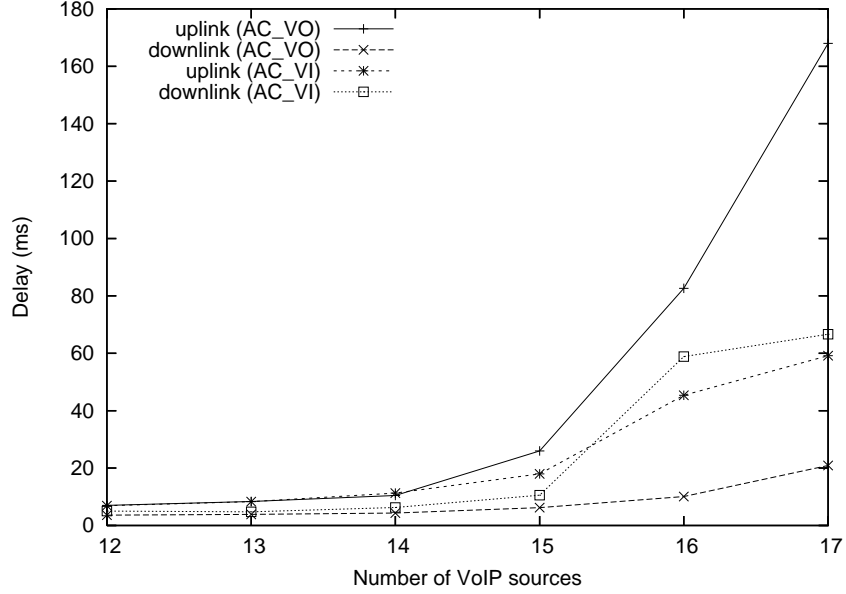


Figure 5.16: 90th percentile of delay of CBR VoIP traffic in AC\_VO and AC\_VI modes of 802.11e

## 5.6 Experimental capacity for VoIP traffic with 802.11e

The Atheros chipset supports Wireless Multimedia Extension (WME) that implements the 802.11e standard, and the capacity for VoIP traffic was measured using the 802.11e feature. The IEEE 802.11e standard was explained in Chapter 1 Section 1.3.5.

### 5.6.1 Capacity for VoIP traffic in IEEE 802.11e

Fig. 5.16 shows the uplink and downlink delay of VoIP traffic in AC\_VO and AC\_VI. When AC\_VO is used for VoIP traffic, the downlink delay remains low, because the AP can transmit multiple frames during its TXOP (transmission opportunity), which is the duration nodes are allowed to transmit frames without contention for, fully utilizing the TXOP. However, the uplink delay increases drastically when the channel becomes congested, because the uplink needs to wait until the AP finishes transmitting frames during its TXOP. While the AP uses the whole TXOP, the uplink does not because a client has only three frames on average even with 17 VoIP calls in the outgoing queue at the start of TXOP, as the average delay was less than 60 ms with 17 VoIP calls. Also, we can see that the capacity is the same as that of DCF regardless of 3 ms TXOP, which reduces the overhead of the backoff time: during 3 ms, six VoIP frames ( $\lfloor 3/0.428 \rfloor^5$ ) can be transmitted, and thus only five of six (83%) downlink frames are transmitted without backoff, theoretically. The low capacity is caused by the significantly increased retry rate, as we can see in Fig. 5.17. The downlink retry rate of AC\_VO is almost 30%, which is six times larger than that of DCF in Fig 5.6, and the uplink retry rate also increases significantly as the number of VoIP sources reaches the capacity. According to additional experiments and analysis, it was found that this happens because of TXOP; when 3 ms TXOP was set with DCF configuration, the retry rate increased in the same way. The reason was that some clients could not detect the transmission of the AP, and they transmitted

<sup>5</sup> $T_t = T_v + SIFS + T_{ACK} + SIFS, T_v = 266\mu s, T_{ACK} = 152\mu s, SIFS = 10\mu s$

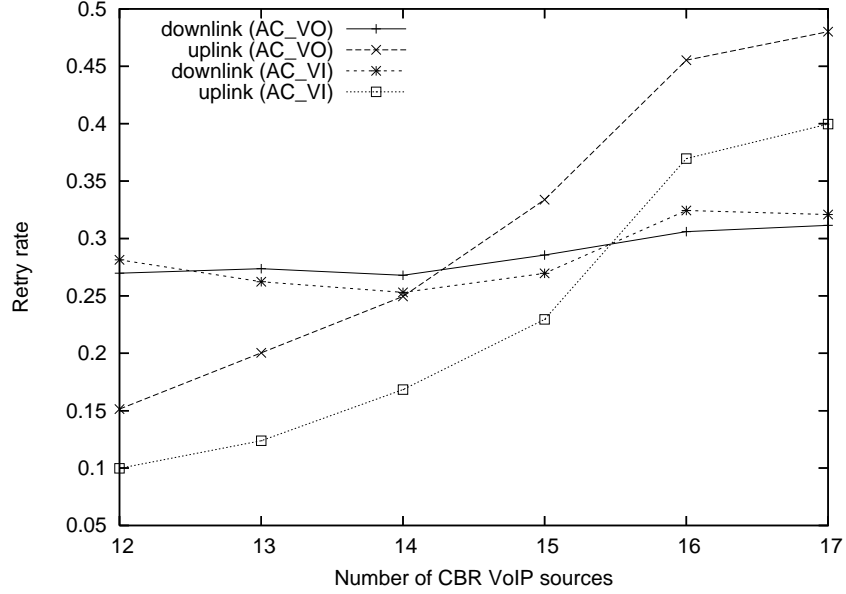


Figure 5.17: Retry rate of CBR VoIP traffic in AC\_VO and AC\_VI in 802.11e

frames during the TXOP of the AP. Even though the distance between the AP and all clients are close enough to detect the transmission of the AP theoretically, some clients could not, in reality. It was actually confirmed in the experiments that the transmission of the AP during its TXOP was interrupted by some clients, by using the second wireless card in the every node as the monitor mode and sniffing all frames. If clients transmit frames during the TXOP of the AP, the frames from clients collide with the frames from the AP because the AP does not detect the medium when transmitting frames during TXOP, following the standard. This causes the significantly higher retry rate in downlink.

In the case of AC\_VI, the capacity increased by only one call even though the TXOP increases to 6 ms, due to the TXOP problem described above.

Figs. 5.18 and 5.19 show the delay and retry rate of VBR VoIP traffic using AC\_VO and AC\_VI. The capacity decreased slightly by one call, because the downlink retry rate increases to 44% with 39 VoIP calls due to the TXOP problem.

### 5.6.2 Effect of TCP traffic on VoIP traffic

In order to identify how the IEEE 802.11e standard can protect the QoS of VoIP traffic against background traffic, the capacity for VoIP traffic was measured with TCP traffic, setting the access category of TCP traffic as AC\_BK. As the traffic generator, the Test TCP (ttcp) utility [79] was used with the default configuration.

Fig. 5.20 shows the results. We can see that the effect of TCP traffic on VoIP traffic is minor: the maximum increase of uplink delay due to TCP traffic is around 50 ms, reducing the capacity by one call. Considering that the capacity of VoIP traffic in DCF decreases to 5 calls when TCP traffic is present according to our experiments, it shows that 802.11e works well as designed.

The commercial 802.11b/g wireless cards that do not support 802.11e features use DCF, whose parameters are the same as those of AC\_BE except AIFS (DCF and AC\_BE use 2 and 3 as AIFSN, respec-



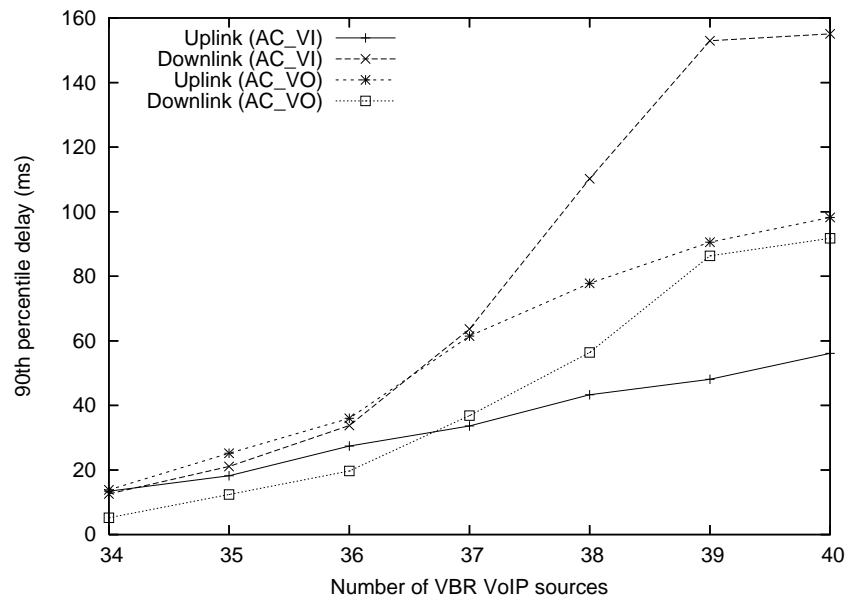


Figure 5.18: 90th percentile of delay of VBR VoIP traffic in each 802.11e AC

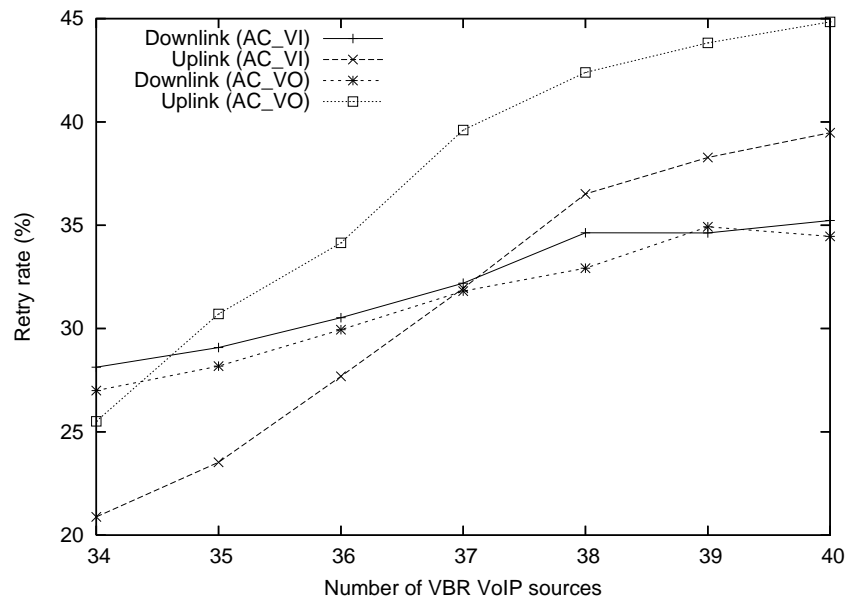


Figure 5.19: Retry rate of VBR VoIP traffic in each 802.11e AC

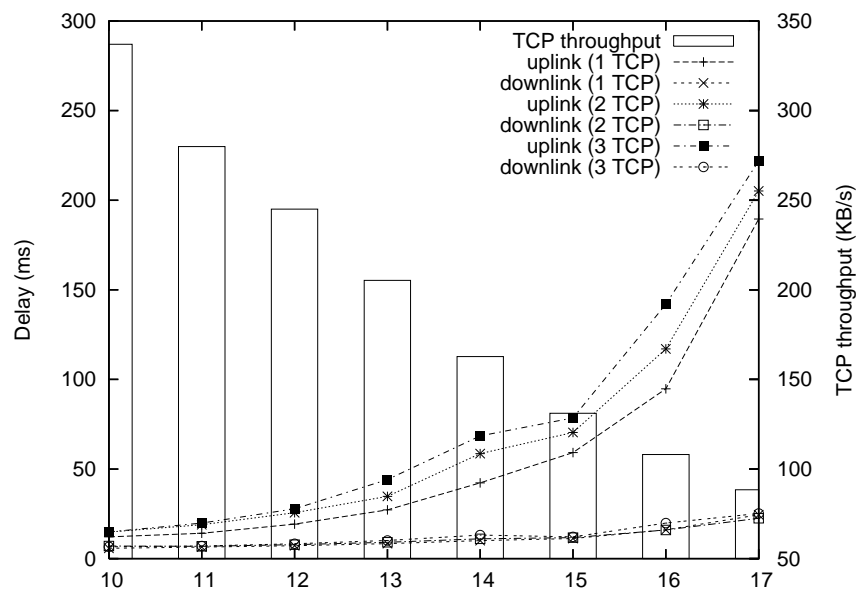


Figure 5.20: Experimental results of CBR VoIP traffic (AC\_VO) with 1 to 3 TCP sources (AC\_BK): 90th percentile of delay of the VoIP traffic and the total throughput of TCP traffic

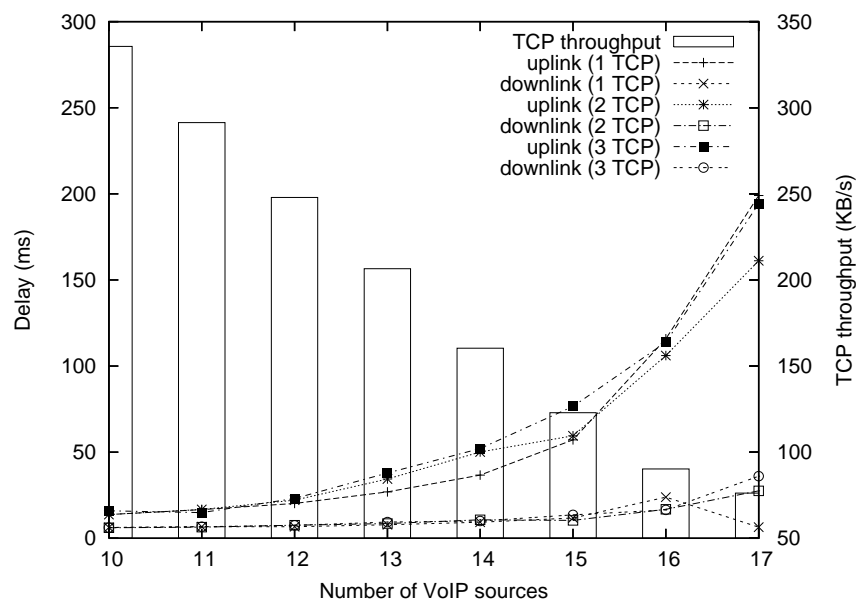


Figure 5.21: Experimental results of CBR VoIP traffic (AC\_VO) with 1 to 3 TCP sources (AC\_BE): 90th percentile of delay of the VoIP traffic and the total throughput of TCP traffic

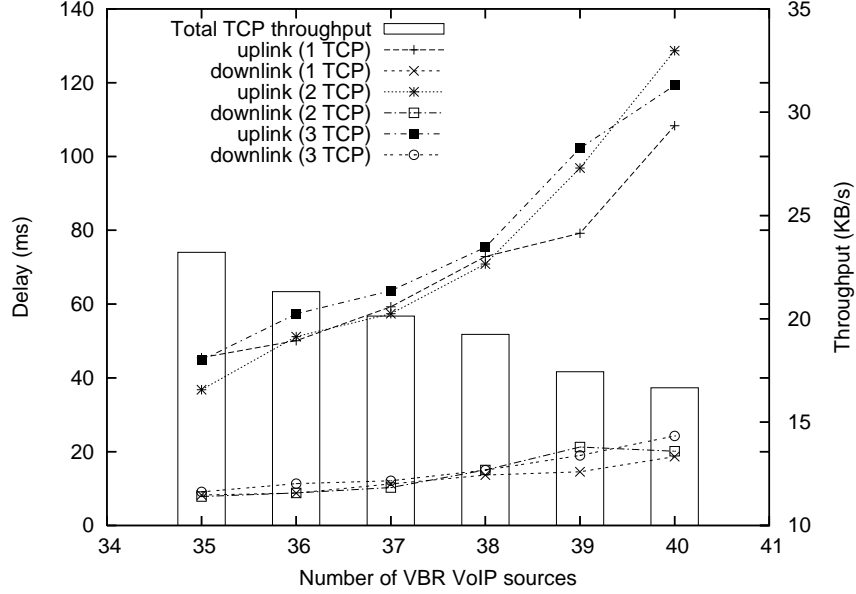


Figure 5.22: Experimental results of VBR VoIP traffic (AC\_VO) with 1 to 3 TCP sources (AC\_BK): 90th percentile of delay of the VoIP traffic and the total throughput of TCP traffic

tively), for any type of traffic, and the 802.11e wireless cards also use AC\_BE for any traffic by default unless applications specify the traffic type. Thus, the same experiments was also performed using AC\_BE for TCP traffic, and Fig. 5.21 shows the results. The total TCP throughput and the delay of VoIP traffic was the same as in the AC\_BK case.

Fig. 5.22 shows the results with VBR VoIP traffic. As in the CBR case, the capacity decreased by only two calls. However, we can notice that the throughput of the TCP traffic is much smaller than that in the CBR case. This is because of the higher retry rate of VBR traffic (refer to Fig. 5.7) and the larger number of VoIP sources.

Moreover, it was found that the number of TCP sources does not affect the QoS for VoIP traffic in both CBR and VBR VoIP traffic. This is because all TCP packets including TCP ACKs use one queue with AC\_BK at the AP, regardless of the number of TCP flows. For this reason, the total TCP throughput was the same regardless of the number of TCP flows. The total throughput in Figs. 5.20 to 5.22 show the average TCP throughput of the three cases.

From the experimental results, we can conclude that QoS of real time traffic like VoIP traffic can be guaranteed well using IEEE 802.11e standard.

### 5.6.3 Impact of each 802.11e parameter

In Section 5.6.1, we saw that using AC\_VO and AC\_VI for VoIP traffic does not increase the capacity due to the TXOP problem. Then, what if we disable the TXOP in AC\_VO? Also, in Section 5.6.2, we saw that the impact of TCP traffic using AC\_BE on VoIP traffic is the same as that using AC\_BK. It could mean that QoS for VoIP traffic can be protected against TCP traffic only by assigning different TXOP values, because the difference between the two ACs is only their AIFSN values (AC\_BE uses 3, and AC\_BK uses 7). In order to investigate the problems, the impact of each 802.11e parameter was identified

Table 5.3: Experimental configuration

Traffic Params	VoIP traffic			TCP traffic			Effective parameter
	CWmin/CWmax	AIFS	TXOP	CWmin/CWmax	AIFS	TXOP	
Config 1	<b>7/15</b>	2	0	<b>31/1024</b>	2	0	CW
Config 2	7/15	<b>2</b>	0	7/15	<b>7</b>	0	AIFS
Config 3	<b>7/15</b>	<b>2</b>	0	<b>31/1024</b>	<b>7</b>	0	CW + AIFS
Config 4	7/15	2	<b>3264</b>	31/1024	2	<b>0</b>	TXOP

via additional experiments. The delay of VoIP traffic (64 kb/s and 20 ms packetization interval) and TCP traffic throughput were measured by setting different values to each parameter in each experiment (Table. 5.3), and Fig. 5.23 and Fig. 5.24 show the results.

We can see that the capacity decreases from 15 calls to 13 calls when VoIP traffic is prioritized with only either CW or AIFS (Fig. 5.23(a) and Fig. 5.23(b)), and when both are applied, the capacity still decreases to 14 calls (Fig. 5.23(c)); the delays with 14 calls in Figs. 5.23(a) and 5.23(b) are below 60 ms, but the packet loss rate is about 5% according to Figs. 5.24(a) and 5.24(b), which does not meet the requirements of the QoS for VoIP traffic explained in Section 5.5.8. In the same way, while the delay with 15 calls in Fig. 5.23(c) is under 60 ms, we can see that the packet loss rate is not acceptable from Fig. 5.24(c). However, prioritizing VoIP traffic using TXOP only can protect the QoS of VoIP traffic against TCP traffic, keeping the capacity to 15 calls, even though the delay slightly increases (Fig. 5.23(d)). Instead, the throughput of TCP traffic decreased; with 15 VoIP sources, TCP throughput decreased from 1.2 Mb/s in other three cases to 0.8 Mb/s by 0.4 Mb/s. However, the total throughput decreases by only 0.3 Mb/s according to Fig. 5.24. This is because the downlink throughput of VoIP traffic using TXOP only is slightly bigger than other cases due to the lower packet loss rate.

## 5.7 Related work

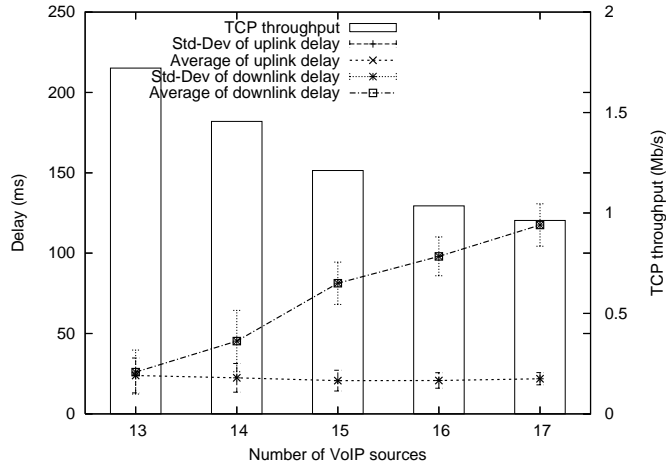
Hole et al. [22] provides an analytical upper bound on the capacity for VoIP applications in IEEE 802.11b networks, evaluating a wide range of scenarios including different delay constraints, channel conditions and voice encoding schemes using analysis, assuming only the long preamble. The capacity of 64 kb/s CBR VoIP traffic with the low bit error rate was the same as my experimental results.

Veeraraghavan et al. [83] analyzed the capacity of a system that uses PCF, where clients can transmit data frames only when they are polled by the AP, for CBR and VBR voice traffic, using Brady's model [6] for VBR voice traffic. In their analysis, they used values of 75 ms and 90 ms as the CFP interval, which causes a delay that is not acceptable for VoIP traffic. The capacity for VoIP traffic with a 90 ms CFP interval was 26 voice calls, but the maximum delay was 303 ms.

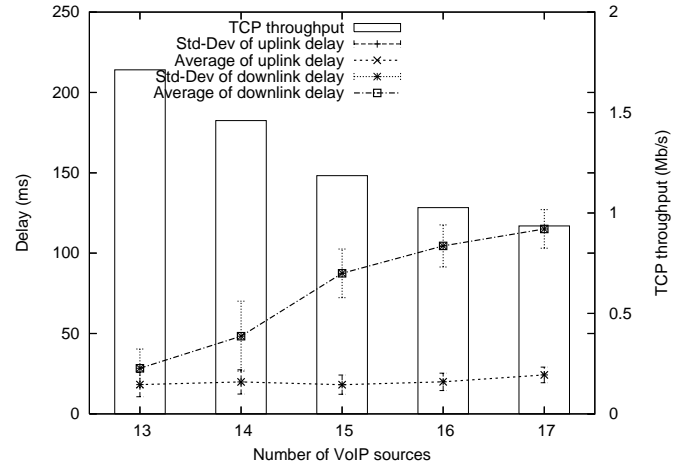
Chen et al. [9] evaluated the capacity of VoIP traffic, via simulations with IEEE 802.11e Enhanced DCF (EDCF) and Enhanced PCF (EPCF), which are called EDCA and HCCA in the 802.11e standard. They used G.711, G.729 and G.723.1 as voice codecs and assumed CBR traffic. IEEE 802.11e provides low end-to-end delay for voice packets even if mixed with best effort traffic.

In [18] and [41], the capacity for VoIP traffic was measured experimentally. However, most of those factors mentioned in the previous section were not taken into account, and no comparison with simulation results was provided.

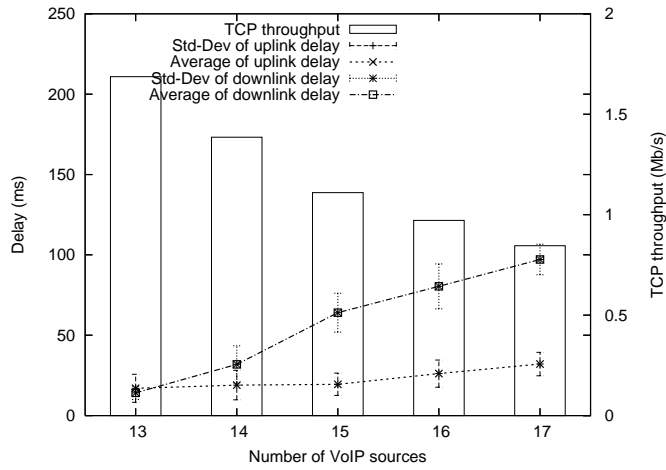
Sachin et. al. [18] experimentally measured the capacity for VoIP traffic with a 10 ms packetization interval and the effect of VoIP traffic on UDP data traffic in 802.11b. They found that the capacity of



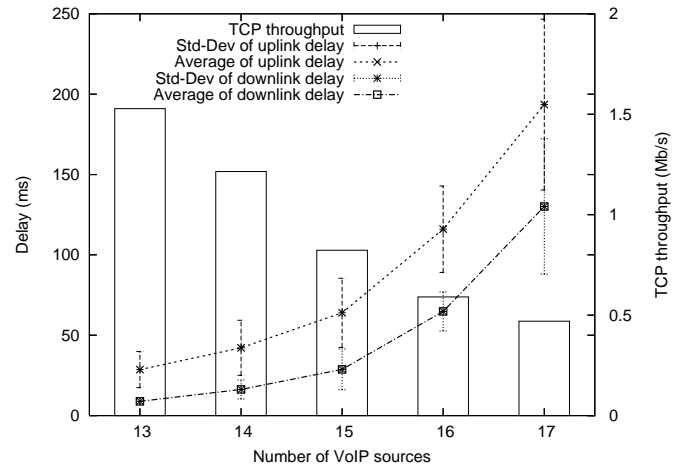
(a) Impact of CW (config 1)



(b) Impact of AIFS (config 2)



(c) Impact of CW+AIFS (config 3)



(d) Impact of TXOP (config 4)

Figure 5.23: The effect of each 802.11e parameter; delay is 90th percentile (refer to Table 5.3 for the experimental parameters in each case.)

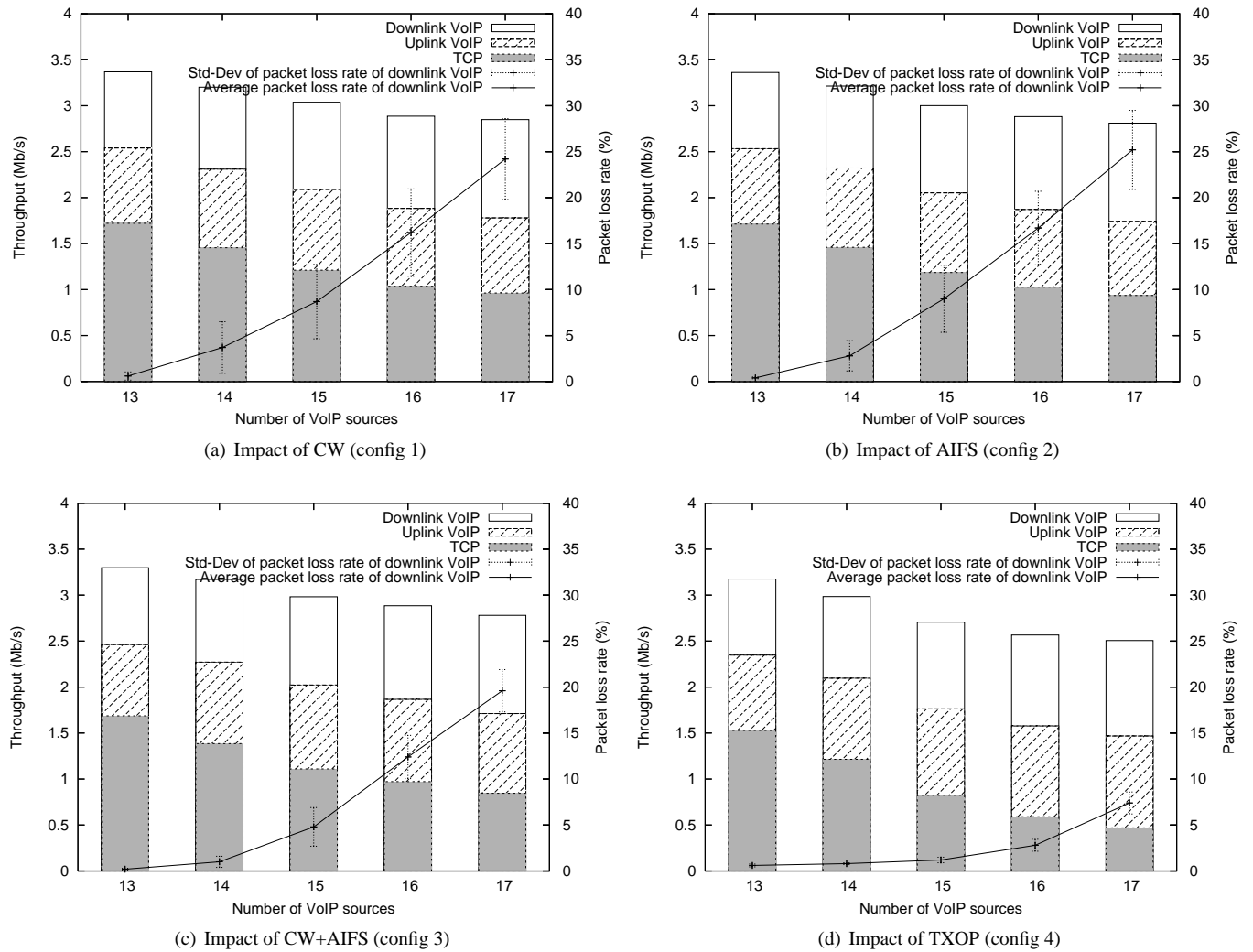


Figure 5.24: The effect of each 802.11e parameter on total throughput and packet loss rate

such VoIP traffic is six calls and the effective available bandwidth is reduced by ongoing VoIP connections.

Anjum et. al. [41] also measured the capacity and the performance of their new scheme, Backoff Control and Priority Queuing (BC-PQ) experimentally. However, in order to determine the capacity for VoIP traffic, they used the packet loss rate, which depends on the network buffer size of the AP in DCF, unless the wireless link is unreliable, as shown in Section 5.5.8. They found that the capacity with 20 ms packetization interval is 10 calls, which differs from our results. We believe that the difference is due to the effect of the Auto Rate Fallback (ARF) and preamble size, but such parameters are not mentioned in the paper.

## 5.8 Conclusion

In this chapter, the capacity for VoIP traffic was measured via experiments with actual wireless clients in the ORBIT test-bed, and compared it with the theoretical capacity and simulation results. Also, some factors were identified that are commonly ignored in simulations and experiments but affect the capacity significantly, and the effect was analyzed in detail with additional experiments and simulations.

Also, it was confirmed that after considering all those factors, we can achieve the same capacity among simulation, experiments, and theoretical analysis, resulting in 15 calls for CBR and 38 calls for VBR VoIP traffic.

The capacity with the 802.11e standard was measured including the effect of TCP traffic on VoIP traffic. From the experiments, it was found that when using 802.11e, the QoS for VoIP traffic is protected well, but the capacity is not improved even with a few milliseconds of TXOP due to significantly increased retransmissions during TXOP.

Even though the effect of those factors on the VoIP capacity was analyzed in this study, those factors affect any experiment and simulation with 802.11 WLANs, and this study can be utilized in their analysis and comparison.

# Chapter 6

## Improving VoIP Capacity in PCF: Dynamic PCF

### 6.1 Introduction

This chapter introduces a new media access schemes based on PCF (Point Coordination Function), Dynamic PCF (DPCF). It improves the capacity for VBR VoIP traffic by up to 30% by minimizing the unnecessary polling and Null Function frames. Also, it has a priority mechanism for VoIP traffic and protects the QoS of VoIP traffic against background traffic like TCP traffic, achieving higher TCP throughput than that of EDCA.

### 6.2 Problems of using PCF for VBR VoIP traffic

In this section, the problems of using PCF for VBR VoIP traffic are identified. As we have reviewed the PCF MAC protocol in Chapter 1, PCF does not suffer from the overhead of collisions and backoff. However, instead, the polling frames can waste some amount of bandwidth in the following situations described in the section below.

#### 6.2.1 Polling during silence period

In full duplex traffic like CBR VoIP traffic, poll bits can be included in downlink data packets from the AP, and polling does not waste any bandwidth. However, in half duplex traffic like VBR VoIP traffic, the AP needs to send CF-Poll frames for clients to send data packets, and if clients are polled when they do not have any data packet to send, they need to send Null function frames, and those frames as well as the unnecessary poll frames waste the bandwidth.

We can compute the bandwidth wasted from the unnecessary polls and Null Function frames using the speech model introduced in Section 5.2.2. In the model, silence period is about 60% of the total conversation time, and on average 60% of 50 polls, that is, 30 polls every second are wasted per client assuming that clients are polled every packetization interval. Considering that a CF-Poll frame and Null function frame is composed of MAC header and the PLCP header, we can easily compute the air time of



those packets assuming 11 Mb/s data rate:  $1 \text{ MAC frame} = 24.73 + 192.0^1 = 216.73 \mu\text{s}$ . Thus, the total air time taken from unnecessary polls and Null Function frames is  $216.73 \times 2 \times 30 = 13 \text{ ms}$  every second per client, which is 143 kb/s ( $13 \text{ ms} \times 11 \text{ Mb/s}$ ) per client. Thus, if the capacity for such VBR VoIP traffic is 30 calls, the total waste of bandwidth is about 4.3 Mb/s. DPCF solves the problem using a dynamic polling list, which will be explained in Section 6.3.1.

### 6.2.2 Synchronization between polls and data

Polls during talking periods can be wasted if the polls are not synchronized with the data frames. As shown in Fig. 6.1(a), if a poll frame arrives before data packet is generated, a null frame is sent from the client, and a pair of a CF-Poll and Null frame just waste the bandwidth. If the packet is sent during the contention period (CP), the next poll is wasted again. Eventually, most of the CF-Polls are wasted and most of the packets are sent during the CP. This is a synchronization problem between CF-Polls and data. If this synchronization problem happens in many nodes, the contention free period (CFP) decreases, CP increases so that the more packets can be sent during CP, and the vicious cycle is repeated, as depicted in Fig. 6.1(b). It was also confirmed via simulations that polls and data are frequently not synchronized, and only a small portion of VoIP packets are sent during CFP. DPCF solves the problem by limiting transmissions of VoIP packets in CP, which will be explained in Section 6.3.2.

### 6.2.3 Multiple packetization intervals

In PCF, the CFP interval should be no longer than the packetization interval to deliver packets without delay. However, VoIP clients can use different packetization intervals ranging from 10 to 50 ms. When more than one packetization interval is used in a wireless network, the choice of the CFP interval affects the capacity and quality of VoIP traffic. For example, we assume that client A and B use 10 ms and 20 ms as their packetization intervals, respectively. In this case, when 10 ms is used as the CFP interval, the client B is polled twice per packetization interval, and a pair of CF-Poll and Null Function frame for the client B wastes bandwidth every 20 ms. When 20 ms is used as the CFP interval, the client A generates 2 packets, but only one packet can be sent during a CFP interval (20 ms) since it is polled only once during a CFP interval. The other packet will be sent in CP or in the next CFP interval, which causes significant delay, in particular, if such delay is accumulated. DPCF solves the problem using the *more data* bit, as we will see in the next section.

## 6.3 Dynamic Point Coordination Function (DPCF)

In this section, we will see how DPCF can solve the problems of PCF mentioned above.

### 6.3.1 DPCF at the AP: Dynamic Polling List

In order to minimize the waste of bandwidth from unnecessary polls during silence periods, the AP manages a dynamic polling list in DPCF. The dynamic polling list is a list of the active nodes only, that is, nodes generating VoIP packets. When a node stops talking, the node is removed from the list, and when it starts to talk, it is added back to the polling list.

---

<sup>1</sup>Long preamble is used.

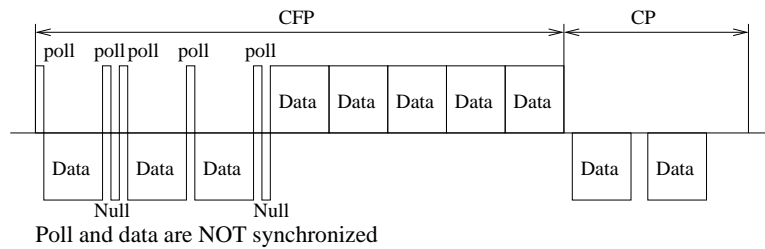
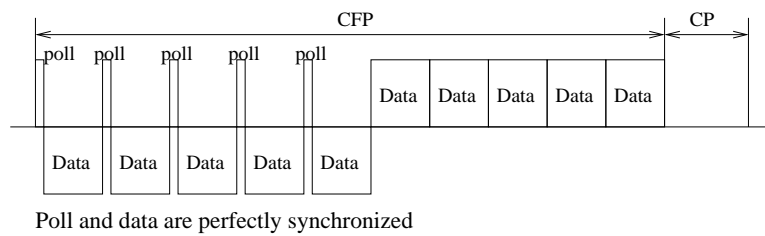
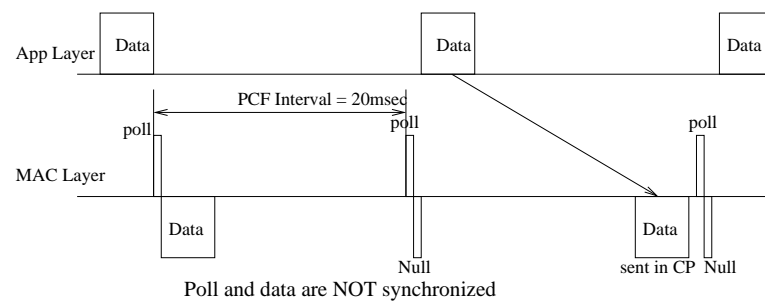
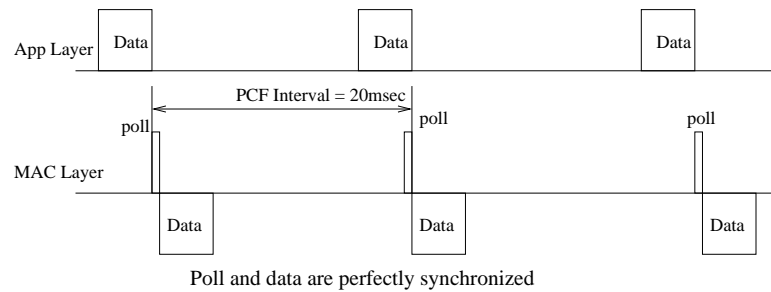


Figure 6.1: Synchronization problem in PCF

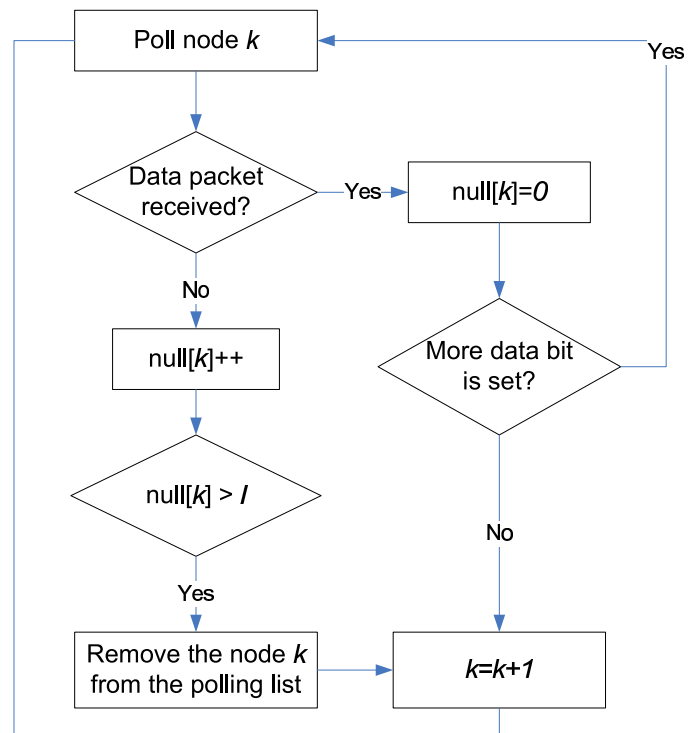


Figure 6.2: DPCF algorithm at the AP side

### Removing a client from the Polling List

When a client stops talking, it sends a Null function frame in response to CF-Polls, and therefore, it needs to be removed from the polling list. However, considering synchronization problem and overhead of putting the node back to the list, the client is removed after the AP receives  $l$  consecutive Null function frames from the client. Generally, once a node is removed from the list, it takes at least two CFP intervals to get a CF-Poll again. Thus, the minimum interval between two CF-Polls is three CFP intervals when a client is removed and added back to the list. Therefore, when the silence period is less than three CFP intervals, it is better to keep the node in the list, and thus, three is used as  $l$  value in DPCF. It was also confirmed via simulations that three is the optimal  $l$  value.

### Adding a node to the Polling List

When the client that was removed from the polling list starts to generate VoIP packets again, the node needs to be inserted to the polling list as soon as possible to avoid delay. One possible approach for adding a node to the polling list is to use statistical properties of VoIP traffic. If we can estimate the duration of the next silence period precisely, the client can be added into the polling list before it starts to send voice packets. Ziouva et al. [94] proposed a scheme where a node can be added into the polling list after  $k$  CFP intervals, with 1 and 2 as  $k$  values. Thus, the statistical approach is tested using ITU-T P.59 [30] for statistical parameters and 500 ms as threshold value for adding a node to the polling list. That is, the client is added to the polling list 1.5 s after it is removed, since the average silence period is 1.5 s in ITU-T P.59. However, because the silence period was exponentially distributed, clients were added either too early or too late, and the bandwidth were wasted with CF-Polls and Null Frames or many VoIP frames were sent during CP.

Therefore, in DPCF, when a client starts to generate VoIP packets, it sends the first VoIP packet during CP. When the AP receives the VoIP packet sent during CP, it adds the client to the polling list, and the client is polled starting from the next CFP. The only problem with this method is that if CP is very congested, the first packet of a talk-spurt can be delayed until the next CP. However, even if the first packet is delayed, the delay is not accumulated because of the *More Data* field; the client will set the field when there are more than one VoIP packet to send, and the AP polls the same client again when the field is set. The *More Data* field will be explained in detail in the next section.

### 6.3.2 DPCF at client side

First, clients need to send a VoIP packet during CP to be inserted in the polling list, as in the standard PCF. After the client is added to the polling list and receives a CF-Poll from the AP, the client can transmit a VoIP packet during CFP. At this time, it can set the *More Data* field if there are more than one VoIP packets to send, so that it can be polled again and transmit the remaining VoIP packets. The *More Data* field is a bit in the 802.11 Frame Control field and is defined in the IEEE 802.11 standard. It is mainly used in power-save mode to indicate that at least one additional buffered MAC Service Data Unit (MSDU) is present at the AP for the same client. Also, the *More Data* field solves the third problem in PCF, explained in Section 6.2.3. In DPCF, when more than one packetization interval is used, the AP uses the largest packetization interval as the CFP interval. The client with smaller packetization interval generally will have two packets a CFP and it just needs to transmit two packets per CFP using *More Data* field. Even though half of the packets from the client will be delayed by the packetization interval, it is not a problem

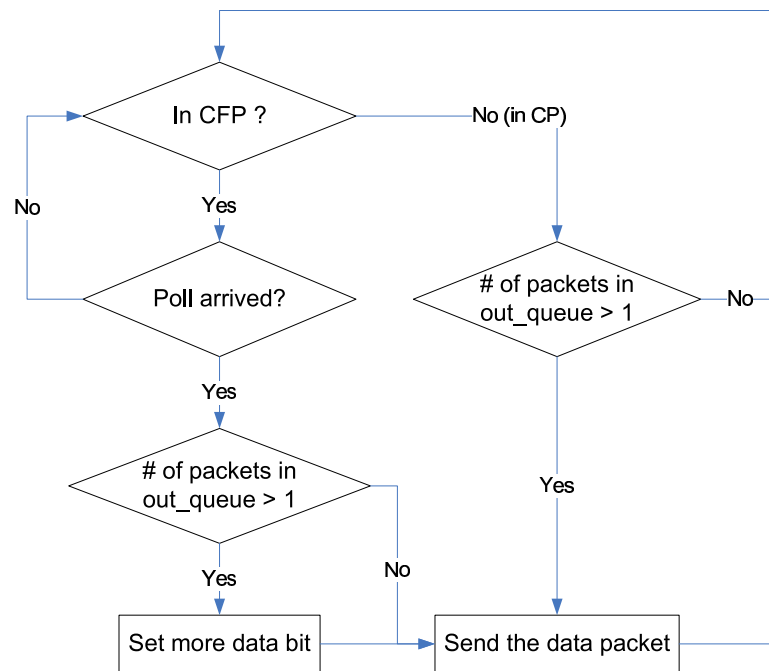


Figure 6.3: DPCF algorithm at the client side

because such delay is neither accumulated nor larger than the packetization interval. Fig. 6.4 shows an example of packet transfer of 10 ms packetization interval VoIP traffic using 20 ms CFP interval in DPCF.

If a VoIP packet is generated after a poll arrives, then the packet needs to be sent during CP. However, DPCF allows clients to send the packet during CP only if they have more than one packet in the outgoing queue. This is to avoid the vicious cycle of asynchronous polling, mentioned in Section 6.2.2. By preventing clients sending the last VoIP packet in the queue during CP, the packet will be sent during CFP when the client is polled. In this way, the poll is not wasted, and thus CFP duration will not shrink. Even though the packet can be delayed slightly, but the delay is smaller than the CFP interval and is not accumulated.

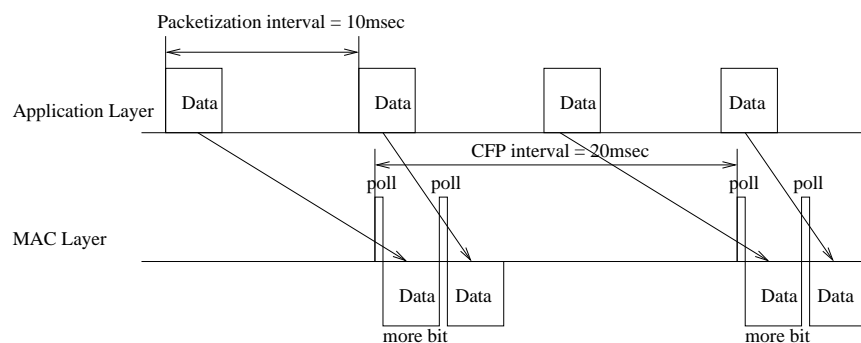


Figure 6.4: Packet transfer in DPCF mode with 20 ms CFP interval

### 6.3.3 Differentiation of traffic types

HCCA has been proposed in the IEEE 802.11e standard [26] to protect the QoS for real time traffic like voice and video traffic against other type of traffic. DPCF is completely compatible with the standard and works on top of the HCCA. However, DPCF itself also provides a way to protect the QoS for VoIP traffic against background traffic by differentiating traffic.

In DPCF, clients transmit VoIP packets during CFP, and also CP if they have more than one packets to send. However, best effort traffic packets must be sent only during CP. This offers higher priority to VoIP packets and reduces the delay due to background traffic. The performance of this method will be evaluated in Section 6.5.2.

## 6.4 Simulations

In order to evaluate the capacity of VoIP traffic, DPCF was implemented in the QualNet simulator [64]. The same network topology and similar network parameters as in Section 5.3 were used. Also, long preamble and 11 Mb/s ACK rate were used in the simulations, as they are default values in the simulator.

## 6.5 Results and analysis

### 6.5.1 Capacity of VoIP traffic

Figs. 6.5 show the average of the 90th percentile of the end-to-end delay of VBR voice packets against the number of wireless VoIP clients in DCF, PCF, and DPCF for a bit rate of 11 Mb/s. We can see that the capacity for VoIP traffic is 28, 30, 37 calls in DCF, PCF, and DPCF, respectively.

DPCF improves the capacity from 30 calls (PCF) to 37 calls by 20%. This is because while PCF wastes bandwidth from unnecessary CF-Polls and Null function frames, DPCF minimizes the number of unnecessary CF-Polls and Null function frames. Fig. 6.6 shows the number of polls and Null function frames in PCF and DPCF with 30 VBR VoIP clients. As we can see, the number of polls decreased to a half in DPCF, and the number of Null function frames decreased to 10%. In the experiment, DPCF eliminated 98,500 unnecessary frames during 130 seconds simulation time, and it corresponds to 56,000 VoIP frames<sup>2</sup>, which is 430 frames per second. Considering that a VoIP call generate 40 packets on average every second with 0.4 activity ratio according to our model, the improvement gain corresponds to about 10 VBR calls, even though actual improvement was 7 calls because additional VoIP calls also increases polls and Null function frames slightly.

### 6.5.2 Capacity of VoIP with data traffic

The performance of DPCF was evaluated with data traffic, namely, TCP connection that runs at the maximum sustainable rate, to see how much VoIP and data interfere with each other in terms of throughput. DCF, PCF, and DPCF were tested with 28 VoIP clients, which is the capacity of VoIP traffic with DCF using the long preamble, and 1 to 3 clients exchanging data traffic.

Figs. 6.7 show the 90th percentile of the end-to-end delay of voice packets and data throughput. We can see that while the end-to-end delay of voice packets in DCF and PCF increases dramatically, the delay in DPCF remains low, as the number of data sessions increases. Furthermore, the figures show that

<sup>2</sup>We convert the number of frames to the total air time of the frames, and the air time is converted to VoIP frames

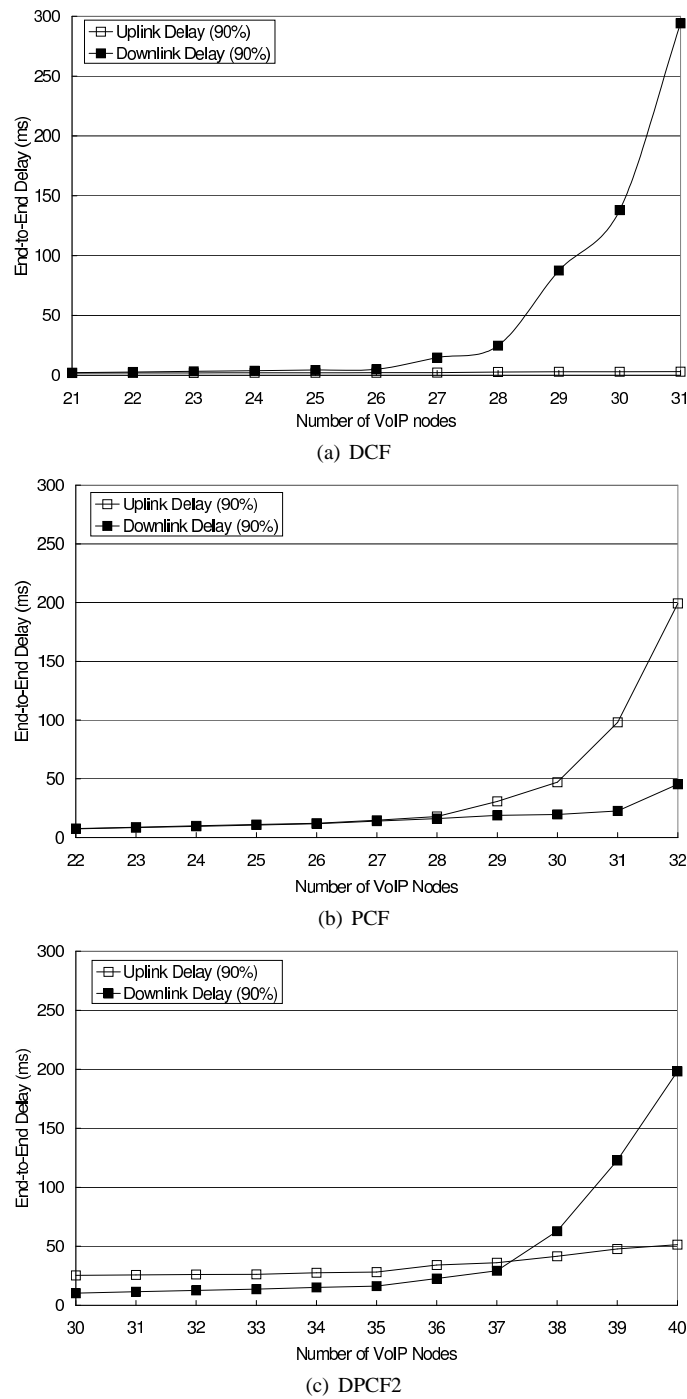


Figure 6.5: 90th percentile of end-to-end delay of VoIP in each MAC protocol

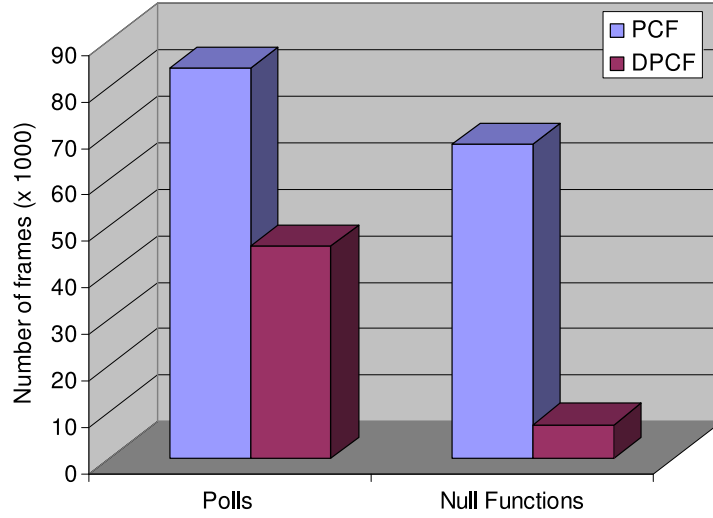


Figure 6.6: Number of polls and Null function frames in PCF and DPCF with 30 VBR VoIP clients

the data throughput is larger in DPCF than those in DCF and PCF; the data throughput is 1.5 Mb/s in DPCF with 3 data traffic, while it is about 600 kb/s in DCF and PCF. DPCF tries to put voice packets into CFP as much as possible in order to reduce the total number of CF-Polls and Null function frames. This also reduces the number of voice packets in CP, allowing other traffic such as data to be transmitted during CP without increasing the end-to-end delay of voice packets.

## 6.6 Comparison with IEEE 802.11e

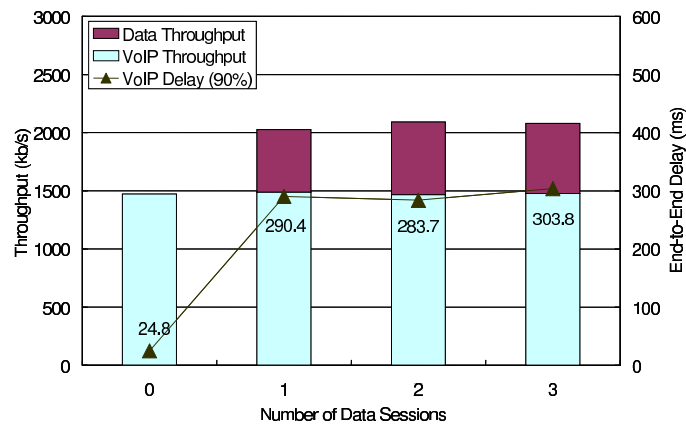
While DPCF can be combined with HCCA to give the higher opportunity to VoIP traffic, DPCF also supports such a differentiation mechanism by itself, as explained in Section 6.3.3; DPCF allows only VoIP packets to be sent in CFP while best effort traffic must be sent only during CP. In this section, we compare the performance of DPCF and EDCA in IEEE 802.11e.

Basically, EDCA does not improve the VoIP capacity because it just gives higher priority to VoIP traffic by using smaller CW, AIFS, and the larger TXOP, as shown in Section 5.6.1 of Chapter 5. Therefore, in terms of the capacity with VoIP traffic only, the performance of DPCF is better than that of EDCA.

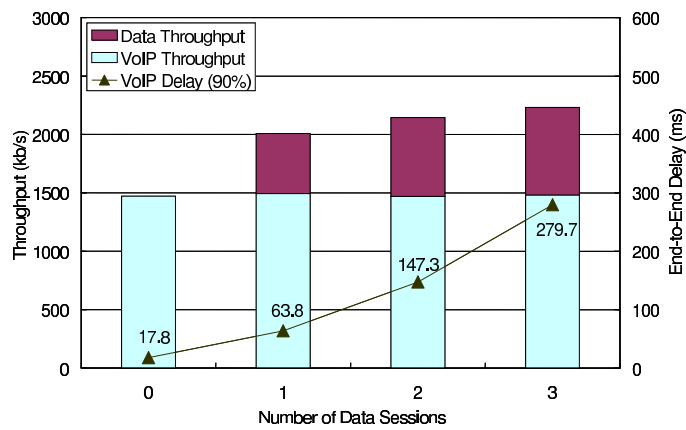
Also, the TCP throughput of DPCF was higher than that in EDCA. The performance of EDCA with TCP traffic was evaluated in NS-2 [48] with TKN EDCA module [88] using the same parameters in simulations of DPCF, using 26 VoIP calls because the capacity of DCF and EDCA was 26 calls in NS-2, and Fig. 6.8 shows the result. We can see that the delay remains low regardless of TCP traffic also in EDCA. However, the TCP throughput is much lower than that of DPCF with 28 VoIP calls<sup>3</sup> and TCP traffic (Fig. 6.7(c)); the TCP throughput in DPCF is 1500 kb/s and that in EDCA is only 500 kb/s. It was also confirmed via simulations that the capacity with TCP traffic in DPCF is 35 calls, which is much larger than that of EDCA, allowing the difference of the two simulators. Therefore, we can conclude that DPCF performs better than EDCA both in the VoIP capacity and throughput of TCP traffic.

<sup>3</sup>I used 28 VoIP calls for comparison because the capacity of DCF in QualNet is 28 calls.

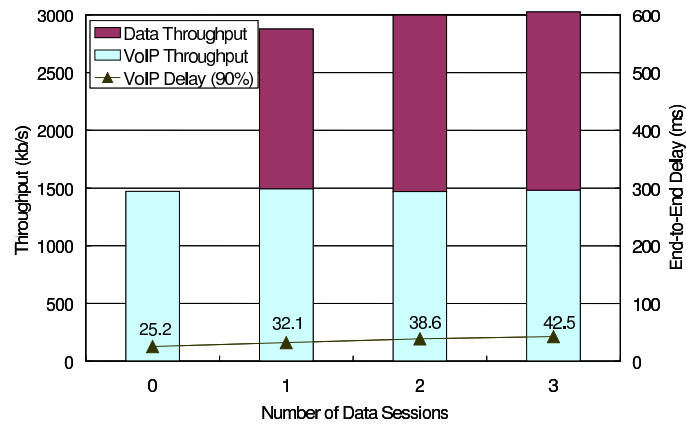




(a) DCF with 28 VoIP nodes



(b) PCF with 28 VoIP nodes



(c) DPCF2 with 28 VoIP nodes

Figure 6.7: 90th percentile of delay of VoIP traffic and throughput of TCP traffic with 28 VoIP clients and 1 to 3 TCP flows

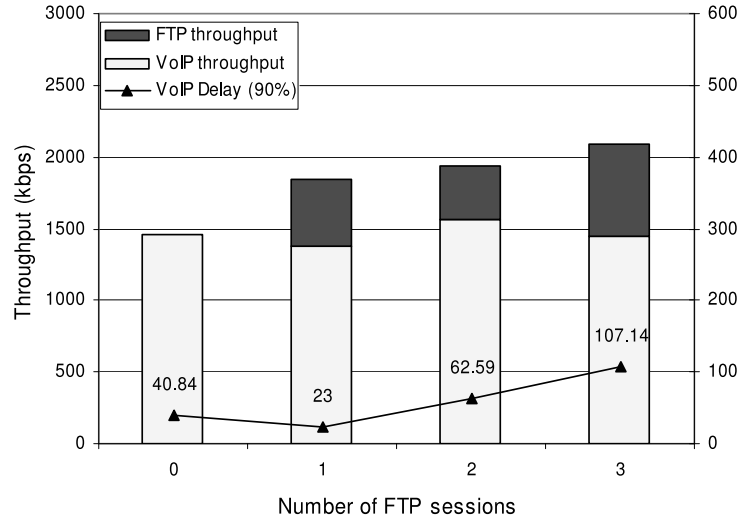


Figure 6.8: Simulation results of EDCA with TCP traffic in NS-2

## 6.7 Related work

Many papers including [75],[21], [92], [86], and [8] proposed methods to improve the VoIP capacity in DCF, and the followings are methods to improve PCF MAC protocol.

Suzuki et al [76] proposed a multiple-polling-list scheme in which VoIP clients are listed in the high-priority list. In their scheme, the PC (Point Coordinator) polls clients in the high-priority polling list first. They used a two-state Markov on/off model for VoIP traffic with exponentially distributed talk-spurts (1 s) and silence periods (1.5 s). Their scheme can reduce the packet loss probability of VoIP traffic from 5% to below 1% with 11 VoIP calls when VoIP and other traffic coexist. However, they did not consider reducing the number of Null function frames.

Yeh et. al.[91] evaluated their various polling schemes of PCF such as Round Robin, FIFO, Priority, and Priority-ELF via simulations, and they evaluated the advantages and disadvantages of each scheme. But they tested only CBR VoIP traffic. Chen et. al. [10] estimated the capacity for VBR VoIP traffic with PCF via simulations, using various voice model such as Brady [6]. However, they they did not identify the overhead of polling during silence periods.

The following papers tried to reduce the polling overhead for VBR VoIP traffic. Kospel et. al. [42] evaluated the performance of DCF and PCF theoretically using a radio channel model. As a part of this study, they also tried to improve the performance of PCF by minimizing the unsuccessful polls. They just assumed that nodes should be removed right after any unsuccessful poll. However, they did not realize that polling can fail because of the synchronization problem between polling and data and did not identify overhead of putting nodes back to the polling list, because they did not perform any simulation for the scheme.

Kim et. al. [40] also proposed an adaptive polling scheme in PCF. Each node notifies the AP of

the empty queue when it sends the last packet in the queue, and the AP removes the node from the list. However, unless the channel is heavily congested, the queue size of each node is mostly one or zero, and nodes will be removed too frequently even during talking periods. They also use a talk-spurt detection algorithm where the AP polls the nodes removed from the list using multiple time intervals, which are determined by a theoretical analysis. Even though they use theoretical analysis to estimate the talk-spurt starting time, it is difficult to predict it, as it is confirmed via simulations, and the bandwidth is wasted due to the polls to check the restart of talk-spurt.

Ziouva et al. [94] presented a new polling scheme called CSSR (Cyclic Shift and Station Removal) for efficient support of VoIP over IEEE 802.11 networks, and improved the capacity by up to 18%. One similarity with DPCF is the use of an “active polling list”. Only active nodes in the active polling list are polled by the AP. However, there are many differences. First, the polling list management scheme is different. In the CSSR polling scheme, a node is removed from the polling list when the start of a silence period is detected and it is added to the polling list  $k$  polling cycles after it is removed. In DPCF, a node is removed when the AP detects three consecutive Null function frames, and a node is added when the AP gets a packet from the node during CP (Section 6.3.1). Secondly, the CSSR polling scheme uses a cyclic shifting of the position of the nodes in the polling list, in order to guarantee a uniformly distributed packet loss among the nodes. This packet loss occurs because if a new packet is generated before the previous packet has been transmitted, the older packet is discarded in the CSSR polling scheme. In DPCF, when a node has more than one packet in its queue, all the pending packets are sent using the More Data field without introducing any additional packet loss. This makes the polling list management scheme in DPCF much simpler than that in CSSR, not requiring any cyclic shift process. Also, the CSSR scheme does not differentiate classes of traffic.

## 6.8 Conclusions

PCF wastes a lot of bandwidth from unnecessary polls and Null function frames in VBR VoIP traffic, and DPCF minimizes the polling overhead by using efficient management of dynamic polling list, the more data field, and synchronization between polls and data packets.

DPCF was implemented using the QualNet simulator, and it was confirmed that DPCF improves the capacity of VoIP traffic by 30%, by decreasing the unnecessary polls to a half and Null function frames to 10%.

DPCF can be combined with 802.11e HCCA, but it has also a mechanism to give a higher priority to VoIP traffic over background traffic. Simulations confirmed that DPCF can protect the QoS of VoIP traffic against TCP traffic, achieving the higher TCP throughput and VoIP capacity than those of 802.11e EDCA.

## Chapter 7

# Improving VoIP Capacity in DCF: Adaptive Priority Control

### 7.1 Introduction

In current IEEE 802.11 wireless networks, as the number of VoIP flows increases, the downlink delay increases while the uplink delay stays very low, as we have seen in Chapter 5 (Fig. 7.1). The reason is that while the AP needs to send more packets than each node, DCF grants the AP and nodes the same chance to send packets. In other words, the delay is unbalanced because resources are distributed unfairly. The unbalanced uplink and downlink delay decrease the VoIP capacity because both uplink and downlink delays need to meet the requirement for QoS of VoIP traffic. Therefore, to increase the VoIP capacity and to reduce the impact of the temporary channel congestion, the uplink and downlink delay need to be better balanced. In this chapter, I introduce Adaptive Priority Control (APC), which adaptively balances the uplink and downlink delay while increasing the capacity for VoIP traffic by roughly 25%.

Section 7.2 describes the algorithm of APC, Section 7.3 analyzes the APC algorithm theoretically, Section 7.4 shows the simulation results, Section 7.5 explains the implementation of APC and the experimental results, Section 7.6 presents the experimental results with IEEE 802.11e, and Section 7.7 describes two methods to implement APC without modifying clients.

### 7.2 Adaptive Priority Control (APC)

The imbalance between uplink and downlink delay is caused by the unfair distribution of channel resources in DCF, as mentioned in the introduction. The uplink and downlink delay are dominated by the queuing delay when the channel is very congested, considering that the transmission and propagation delay in IEEE 802.11 wireless networks are very small compared with the queuing delay. Also, the queue size of the AP is much larger than that of wireless clients with a large number of VoIP sources, because the AP receives all packets to all the wireless clients. Thus, the queuing delay at the AP is also much bigger than those of the wireless clients, which causes difference between uplink and downlink delays. Therefore, the AP needs to be given more chances to transmit frames; APC increases the priority of the AP relative to that of the wireless clients adaptively according to wireless channel conditions and the uplink and downlink traffic volume.

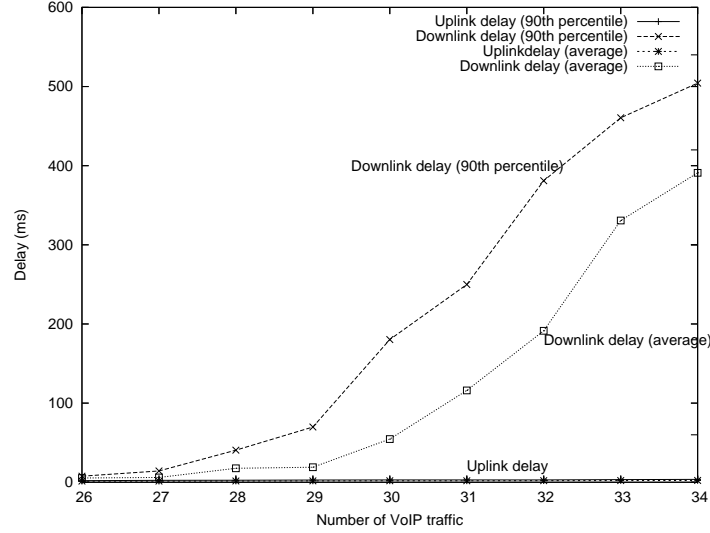


Figure 7.1: The uplink and downlink delay of VoIP traffic in DCF

### 7.2.1 Priority mechanisms at the MAC layer

Before we see how to set the optimal priority of the AP to balance the uplink and downlink delay, we will see how to grant the priority to the AP at the MAC layer. In IEEE 802.11, there are three well-known methods to control the priority of wireless nodes, namely, control contention window, control interframe space, and burst frames. All three methods are used in IEEE 802.11e to differentiate the priorities of frames according to the Access Category (AC).

#### Control contention window (CW)

The first method controls contention window (CW) size. The backoff time of a frame is chosen randomly between 0 and CW measured in slots. When nodes have a smaller window size, the backoff time decreases and the transmission rate increases. However, this method increases the collision rate as the window size decreases [56], and that it is difficult to accurately control the priority since the backoff time is chosen randomly within the CW size. The increased collision rate significantly reduces throughput, as shown in Section 7.4.2.

#### Control InterFrame Space (IFS)

The second method changes the Inter-Frame Spacing (IFS). The node with the smaller IFS has the higher chance to gain access to the channel when two nodes are trying to transmit frames simultaneously. However, we cannot accurately control the transmission rate using this method because the backoff time is still decided randomly, as in the first method.

#### Contention Free Transmission (CFT)

The last method transmits multiple frames contention free, i.e., without backoff, when a node gets a chance to transmit a frame. The IEEE 802.11e standard [26] proposes this, calling it Contention Free Burst (CFB).

However, in CFB the frames are sent contention free for a fixed amount of time, called Transmission Opportunity (TXOP). I propose a variant called CFT, where the number of frames to be sent contention free changes in real time according to the dynamically changing priority of the node.

APC uses CFT because it allows us to control the transmission rate precisely according to the priority without collision overhead; every node including the AP has the same chance to transmit frames on average in IEEE 802.11 [23]. Thus, if the AP sends  $P$  frames contention free when it gets a chance to transmit a frame, then the AP has exactly  $P$  times higher priority than other wireless clients.

### 7.2.2 APC algorithm

To achieve the fairness between the downlink (the AP) and uplink (wireless clients) in a BSS, when uplink and downlink have the same amount of traffic, the AP needs to be able to send the same number of packets as the total number of packets that all wireless clients send within a given interval. Then, intuitively, the AP needs to send  $N$  frames while  $N$  wireless clients transmit one frame each. I call this 'semi-adaptive priority control (sAPC)' because it is adaptive to the change in the number of the active wireless clients only. In VoIP traffic, when the same packetization interval is used for all VoIP traffic in a BSS, the uplink and downlink traffic volumes are symmetric, with large number of VoIP sources, and thus sAPC would balance the uplink and downlink delay in the case. However, when more than one packetization interval or codec is used for VoIP traffic in a BSS, the traffic volume of the uplink and downlink becomes asymmetric: even when the number of active wireless nodes and wired nodes are the same, the number of packets from the wireless nodes and the wired nodes depends on the packetization intervals of the active nodes. For example, when 10 wired nodes with 10 ms packetization interval and 10 wireless nodes with 20 ms packetization interval are talking at the same 64 kb/s voice bit rate, the volume of the downlink traffic from wired nodes is larger than the uplink traffic volume because of the overhead to transmit a VoIP packet such as MAC/PHY headers. In such a case, we need to consider the traffic volume of uplink and downlink in deciding the priority of the AP.

In order to consider such traffic volume changes, APC uses the ratio of the number of packets in the queue (queue size) of the AP and an average queue size of all wireless clients as the priority of the AP ( $P$ ) when the queue of wireless clients is not empty, and the number of active wireless clients when the queue of clients is empty. That is,  $P$  is calculated as follows:

$$P = \begin{cases} \lceil \frac{Q_{AP}}{Q_C} \rceil & \text{if } Q_C \geq 1 \\ N_e & \text{if } Q_C < 1 \end{cases} \quad (7.1)$$

where,  $Q_{AP}$  is the queue size of the AP,  $Q_C$  is the average queue size of the wireless clients, and  $N_e$  is the number of active wired nodes.

For instance (Fig. 7.2), if four wireless clients, from C1 to C4, have two packets in each queue, and the AP has six packets in its queue, then the average queue size of the wireless clients ( $Q_C$ ) is 2, and the priority of the AP ( $Q_{AP}$ ) becomes 3 ( $=6/2$ ). Thus, in APC, the AP sends three frames contention free when it acquires a chance to transmit a frame. If we assume that every node gets the same chance to transmit frames, then the average number of packets in the queue of the wireless clients and the one of the AP become one and three, respectively, and both of them become zero after the next transmission. Therefore, transmitting  $Q_{AP}/Q_C$  packets contention free results in the same packet processing time in the AP and wireless clients, which means that the AP and wireless clients have the same queuing delay. This is proved for the general case in the next section.

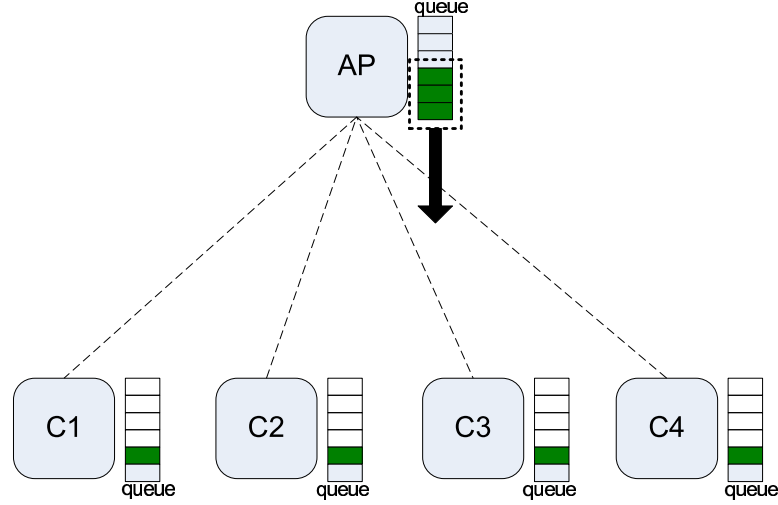


Figure 7.2: Packet transmission in APC

Using this metric, the priority of the AP changes adaptively when the traffic volume of the uplink and downlink changes. When the amount of traffic to the AP increases, the queue size of the AP increases and the priority also increases to balance the downlink delay with the uplink delay. When the queue size of the clients increases, the priority of the AP decreases.

Instead of using the number of packets, we could also use the packet size to compute the ratio between the uplink and downlink traffic volume. However, for our application, the overhead to transmit a voice packet is very large compared to the small voice data size. It was already shown in Chapter 5 that the voice data takes up only 18% of the total VoIP frame size in DCF. It was also confirmed via simulations that using the number of packets queued performs better than using the packet size.

### 7.3 Theoretical analysis

In this section, it is proven that the APC algorithm (Eqn. 7.1) balances the uplink and downlink delay.

The symbols used in the analysis are defined as follows:

$\Delta Q_{AP}$  = Change of the number of packets in the queue of the AP for a second

$Q_{AP}$  = Number of packets in the queue of the AP

$Q_C$  = Average number of packets in the queue of all wireless clients

$D_{AP}$  = Queuing delay of the AP (second)

$D_C$  = Queuing delay of a client

$N_e$  = Number of active (talking) wired nodes

$x_{AP}$  = Transmission overhead (backoff, deferral, and retry) at the AP (second)

$I$  = Packetization interval (second)

$T_v$  = Transmission time of one VoIP frame including ACK

$\lambda$  = Packet arrival rate

$\mu$  = Packet transmission rate

$P$  = Priority of the AP to balance the uplink and downlink delay

The dominant component of delay is the queuing delay considering that the transmission delay and the transmission overhead are very small. Furthermore, the transmission delay is the same in the AP and wireless clients assuming that they use the same transmission rate. The transmission overhead, which includes backoff, deferral and retransmissions, is also similar for the AP and wireless clients, while the queuing delay of the AP is much larger than those of the wireless clients. Therefore, balancing the queuing delay of the AP and wireless clients results in the balanced uplink and downlink delay. Thus, it is shown that APC balances the queuing delay of the AP and wireless clients.

We can compute the queuing delay by multiplying the transmission time by the queue size according to Little's law ( $D_{system} = Q_{system}/\mu_{system}$ ). Then, we can compute the queuing delay of the AP ( $D_{AP}$ ) and the clients ( $D_C$ ) as follows:

$$D_{AP} = Q_{AP} \cdot \frac{1}{\mu_{AP}}$$

$$D_C = Q_C \cdot \frac{1}{\mu_C}$$

We consider the priority of the AP ( $P$ ) in two cases: When the queue size of clients is greater than or equal to 1 ( $Q_C \geq 1$ ) and when the queue size of clients is less than 1 ( $Q_C < 1$ ).

### 7.3.1 Non-empty client queues ( $Q_C \geq 1$ )

When all wireless nodes including the AP have packets to transmit, every wireless node as well as the AP has the same chance to transmit packets due to the fairness of CSMA/CA on average, that is,  $\mu_{AP} = \mu_C$  in DCF. Then, in APC,  $\mu_{AP} = P \cdot \mu_C$  because the AP transmits  $P$  packets when it gets a chance to transmit packets while each client transmits only one packet. Thus,  $D_{AP}$  can be rewritten as:

$$D_{AP} = Q_{AP} \cdot \frac{1}{P \cdot \mu_C}$$

Then, we can get the optimal  $P$  value for balancing the delay of wireless clients and the AP as follows:

$$D_{AP} = D_C$$

$$Q_{AP} \cdot \frac{1}{P \cdot \mu_C} = Q_C \cdot \frac{1}{\mu_C}$$

Then,

$$P = \frac{Q_{AP}}{Q_C}.$$

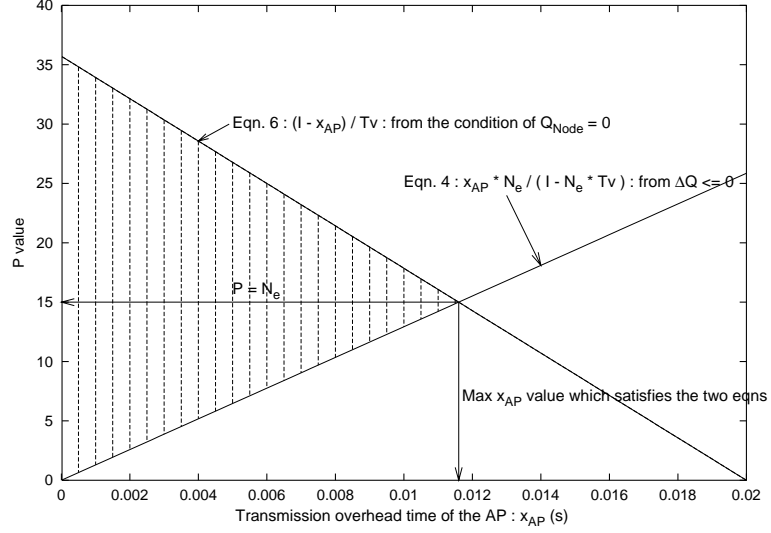
### 7.3.2 Empty client queues $Q_C = 0$

The queue size of wireless clients decreases when the transmission rate of clients at the MAC layer is bigger than the packet generation rate at the application layer, that is  $\mu_C \geq 1/I$ , which is always satisfied if  $Q_C = 0$ . In order to bring the queuing delay of the AP down to zero, the change of the queue size of the AP needs to be less than or equal to zero ( $\Delta Q_{AP} \leq 0$ ). In order to get the priority value of the AP ( $P$ ) that satisfies it, we derive the equation for  $\Delta Q_{AP}$ .

The change of the number of packets in the queue of the AP ( $\Delta Q_{AP}$ ) is the packet arrival rate to the AP minus the packet transmission rate from the AP:

$$\Delta Q_{AP} = \lambda_{AP} - \mu_{AP}$$



Figure 7.3: Optimal P value when  $Q_C = 0$ 

When the AP sends  $P$  packets contention free, the transmission time of a packet is  $(x_{AP} + T_v \cdot P)/P$ , and transmission rate ( $\mu_{AP}$ ) becomes  $P/(x_{AP} + T_v \cdot P)$ . Then,  $\Delta Q_{AP}$  is rewritten as follows:

$$\Delta Q_{AP} = \frac{N_e}{I} - \frac{P}{x_{AP} + T_v \cdot P} \quad (7.2)$$

Here, for  $\Delta Q_{AP} \leq 0$ ,

$$\frac{N_e}{I} \leq \frac{P}{x_{AP} + T_v \cdot P} \quad (7.3)$$

$$P \geq \frac{N_e \cdot x_{AP}}{I - N_e \cdot T_v} \quad (7.4)$$

According to Eqn. 7.4,  $P$  value is proportional to the transmission overhead of the AP as shown in Fig. 7.3; the longer the AP takes time to get access to media, the more packets the AP needs to transmit contention free.

Here, the transmission time of the AP should not exceed the VoIP packetization interval because the wireless clients need to send at least a packet within a packetization interval to keep their queues empty. That is,

$$x_{AP} + T_v \cdot P < I \quad (7.5)$$

Then,

$$P < \frac{I - x_{AP}}{T_v} \quad (7.6)$$

Eqns 7.4 and 7.6 are plotted in Fig. 7.3 with  $T_v = 560 \mu s$ <sup>1</sup>,  $N_e = 15$  and  $I = 20 ms$ , and the shaded region represents the one that satisfies Eqn. 7.4 and Eqn. 7.6. According to the two graphs in Fig. 7.3, we can see that  $P$  should be less than or equal to  $N_e$ . We can also get the same result when we combine Eqn. 7.2 and Eqn. 7.5:

$$\frac{N_e}{I} = \frac{P}{x_{AP} + T_v \cdot P} \geq \frac{P}{I}$$

<sup>1</sup>The  $T_v$  value is calculated with 160B (20 ms packetization interval and G.711 codec) payload in 11 Mb/s transmission rate

$$P \leq N_e$$

Therefore, the optimal  $P$  value that satisfies the two equations in any possible  $x_{AP}$  value is  $N_e$ .

## 7.4 Simulation results

In order to evaluate the performance of APC, APC and sAPC were implemented in the QualNet simulator [64], and the uplink and downlink delay were measured with various packetization intervals. The simulation parameters were those used in Section 6.4, that is, an Ethernet-to-wireless network topology in IEEE 802.11b with 64 kb/s VBR VoIP traffic.

### 7.4.1 Evaluation of APC

#### VoIP traffic using single packetization interval

The graphs in Fig. 7.4 show the simulation results for a 20 ms packetization interval and 64 kb/s VoIP traffic. Both the 90th percentile and average value of uplink and downlink delay were plotted because the 90th percentile value is a good measure of the capacity for the VoIP traffic, and the average value is used to check the balance of the uplink and downlink delay. According to graphs, even though APC performs slightly better, both sAPC and APC balance the uplink and downlink delay effectively when all VoIP sources use the same packetization interval. If we compare those with the results for DCF (Fig. 7.1), we can see that APC improves not only the balance between uplink and downlink delay, but also the capacity for the VoIP traffic by 25%, from 28 calls to 35 calls.

#### VoIP traffic with mixed packetization intervals

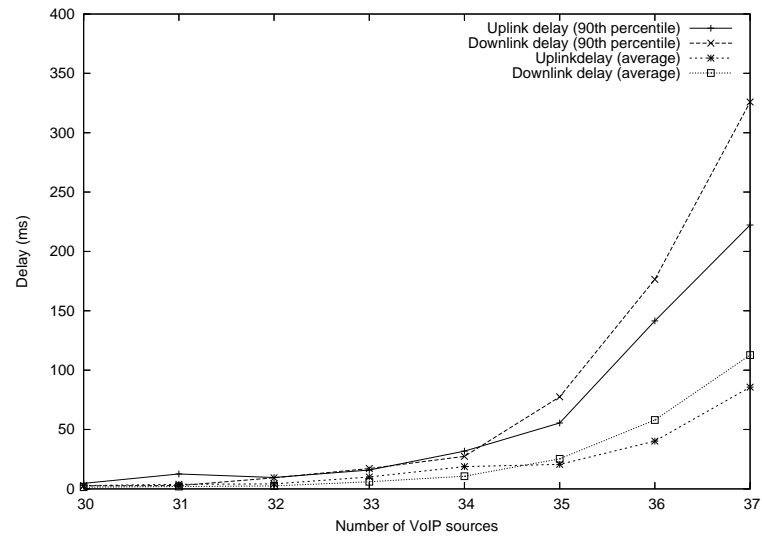
Figs. 7.5 show the simulation results when half the nodes use 10 ms and the other half use 20 ms as their packetization intervals. We can see that the uplink and downlink delay are unbalanced in sAPC as the number of VoIP sources increases, while the two components are still balanced in APC. This is because when more than one packetization interval is used, the traffic volume of uplink and downlink diverge, and APC changes the priority of the AP adaptively to the change of the uplink and downlink traffic volume, while sAPC adapts only to the change of the number of active wireless nodes.

#### VoIP traffic with larger packetization intervals

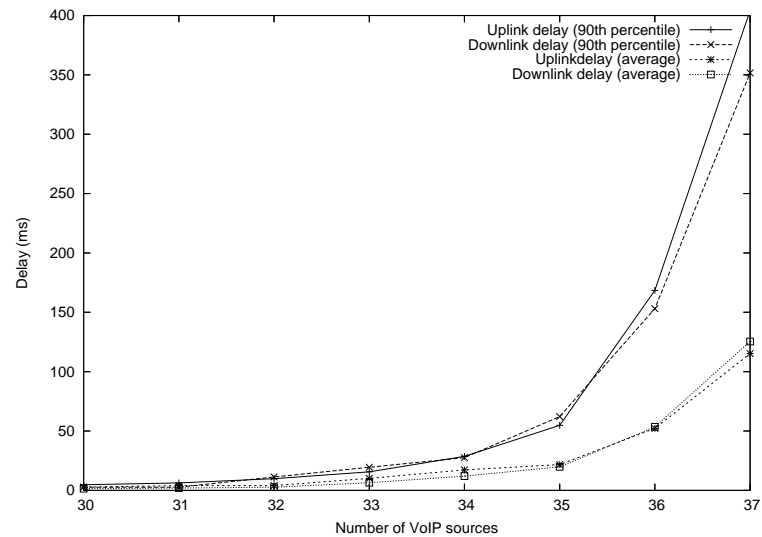
In order to verify the performance across a variety of patterns, the performance of APC was evaluated with other packetization intervals, because its impact on QoS is larger than that of the packet size change. Fig. 7.6 shows the simulation results for APC using VoIP traffic having only 40 ms packetization interval, and Fig. 7.7 shows the simulation results for VoIP traffic equally divided between 20 ms and 40 ms packetization intervals. The two figures illustrate that APC works for various types of VoIP traffic.

#### Instant delay of VoIP traffic in APC

In order to see how the uplink and downlink delays change as a function of simulation time, the two components were plotted throughout the simulation time, and Fig. 7.8 shows a sample simulation result with 36 VBR VoIP sources (64 kb/s and 20 ms packetization interval). We can see that uplink and downlink delay are balanced throughout the whole simulation.

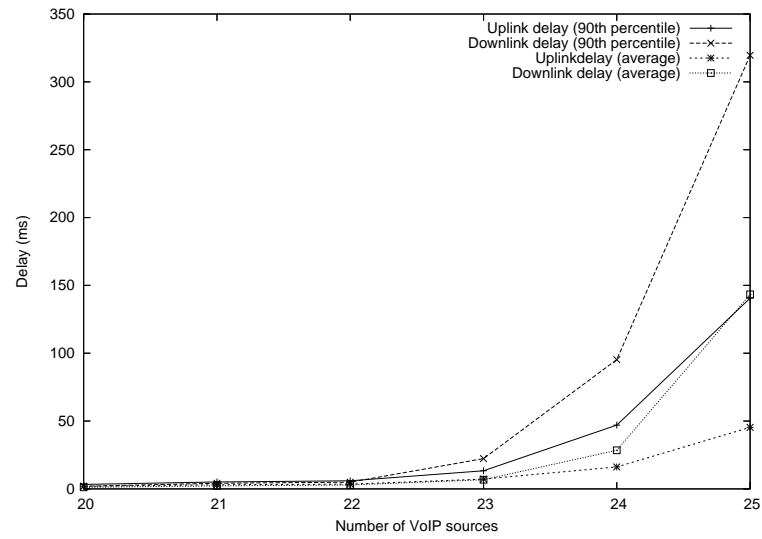


(a) sAPC

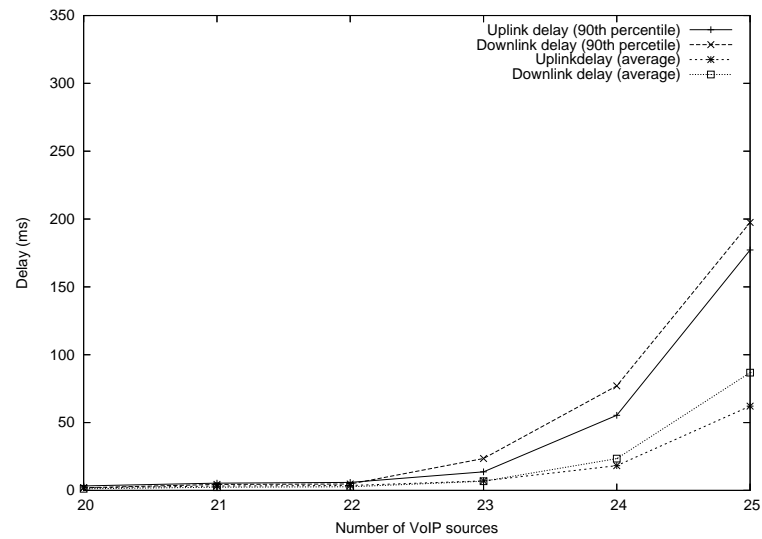


(b) APC

Figure 7.4: Delay as a function of the number of VoIP sources using two priority control methods with 20 ms packetization interval 64 kb/s VoIP traffic



(a) sAPC



(b) APC

Figure 7.5: Delay as a function of the number of VoIP sources using two priority control methods with mixed (10 ms and 20 ms) packetization interval 64 kb/s VBR VoIP traffic

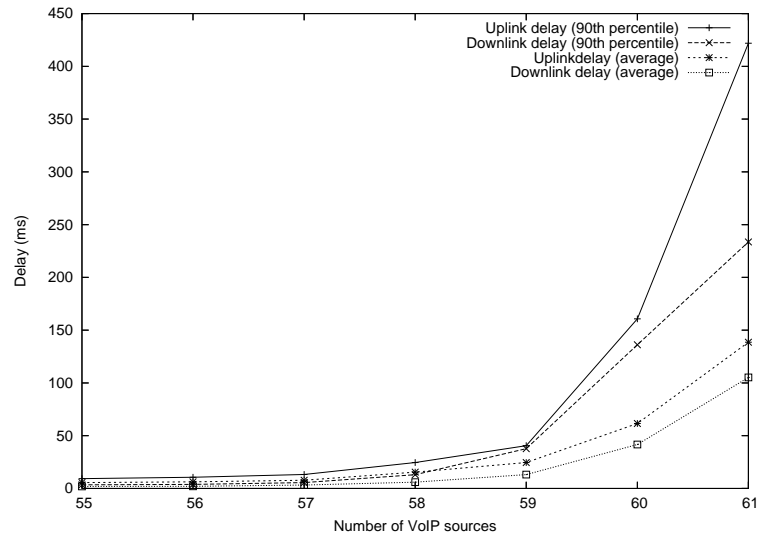


Figure 7.6: Delay as a function of the number of VoIP sources using VoIP traffic with 40 ms packetization intervals

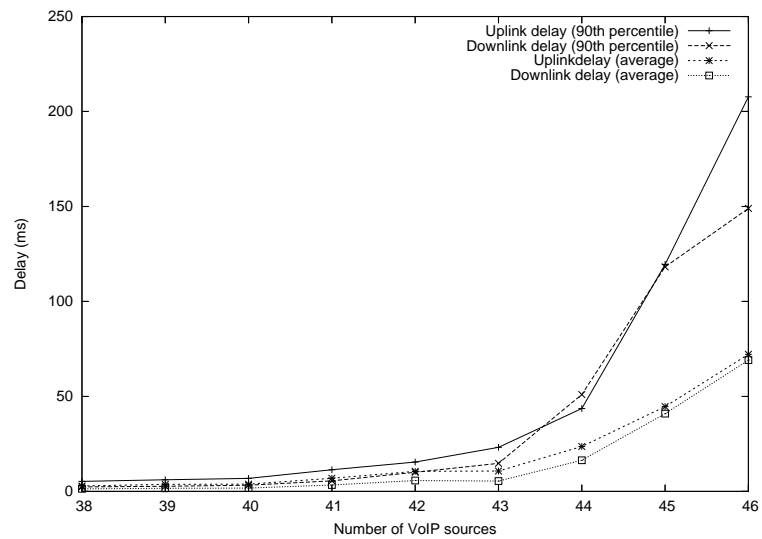


Figure 7.7: Delay as a function of the number of VoIP sources using VoIP traffic with 20 ms and 40 ms packetization intervals

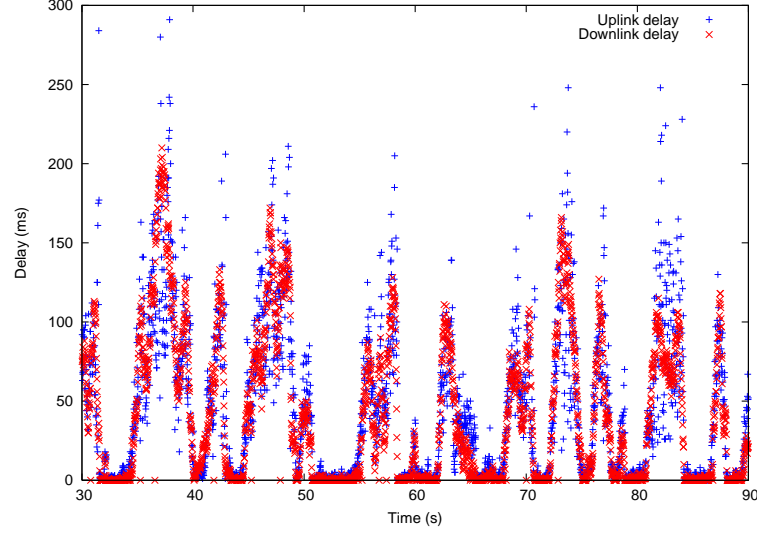


Figure 7.8: The uplink and downlink delay as a function of simulation time with 36 VBR VoIP sources using APC

#### 7.4.2 Comparison with CW control method

Section 7.2.1 mentioned that many papers change CW to control the priority or the transmission rate of the traffic, but it has the overhead that the collision rate increases. In order to verify it, the approach was implemented and the performance was measured.

Fig. 7.9 shows the results of using CW to control the transmission rate. The priority of the AP was calculated using the same algorithm as for APC (Eqn. 7.1 in Section 7.2), and the priority ( $P$ ) was converted to the transmission of the AP as follows.

$$CW = \max(CW_{MIN}/P, 1),$$

where  $P \geq 1$  from Eqn. 7.1,  $1 \leq CW \leq CW_{MIN}$ , and the  $CW_{MIN}$  is the minimum contention window size defined in IEEE 802.11. When the priority of the AP is very high ( $P \geq CW_{MIN}$ ), CW decreases to 1, and the AP transmits packets almost without backoff. When the priority is 1, the CW becomes  $CW_{MIN}$ , and the AP has the same transmission rate as the wireless clients. As shown in Fig. 7.9, although the balance improves somewhat compared with DCF, the uplink delay is much larger than the downlink delay, which means that the AP is given too high a priority. The reason is that even if the CW of the AP is changed to  $1/P$  of  $CW_{MIN}$ , the transmission rate of the AP is not exactly  $P$  times because the backoff time is chosen randomly within the CW size. Another problem of this approach is the high retry rate, as shown in Fig. 7.10. We can see that the retry rate of the AP in CW control approach increases significantly as the number of VoIP sources increases, while APC maintains the same retry rate as the number of VoIP sources increases. This is because using a smaller CW with many wireless nodes increases the collision probability in DCF. The reason why the retry rate of the AP in APC is lower than that in DCF is that contention free transmission of the AP decreases the probability of packet collision.

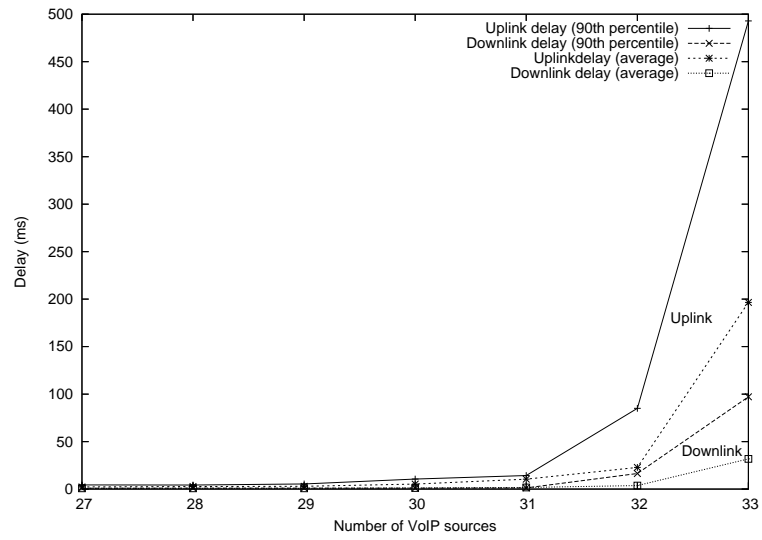


Figure 7.9: Delay as a function of the number of VoIP sources using CW to control the transmission rate

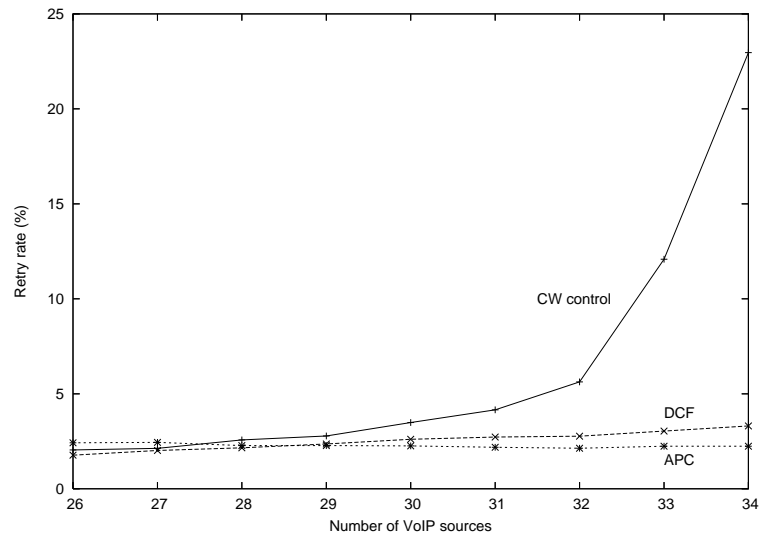


Figure 7.10: Retry rate of the AP (downlink traffic) as a function of the number of VoIP sources in three approaches: Controlling CW, DCF, and APC

## 7.5 Implementation and experiments

The APC algorithm was implemented in a wireless card driver and the performance was measured in the ORBIT test-bed.

### 7.5.1 Implementation

#### Sending client queue size

To report the queue size of each client to the AP, the first two bytes of the frame body was used as the queue size field. In clients, the driver added the two bytes of the current queue size at network layer to the body when constructing the MAC layer packet, when the driver received the data packets from kernel. The AP extracted the first two bytes in the body and used it to compute the  $P$  value when it receives the packets from the clients, and the rest of the body frame was sent to kernel.

#### Implementation of CFT

Ideally, the CFT needs to be implemented in wireless card firmware and the driver just need to be able to set the  $P$  value, but CFT was emulated in the driver because we cannot modify the firmware.

Thus, to implement CFT, the CW and AIFS values were changed. When the driver successfully transmits a packet using the normal backoff procedure, it sets  $CW_{MIN}$  and  $CW_{MAX}$  values to zero and AIFSN value to 1 so that the following packets can be sent contention free, and when the number of packets sent contention free becomes  $P$  value, it sets the CW and AIFS values back to the normal values. TXOP in IEEE 802.11e can be used to emulate CFT, but it was not used because the Atheros chipset does not allow the driver to control the TXOP duration dynamically.

The problem of this implementation method is that the accuracy drops slightly, because the firmware takes time to notify the driver of the completion of transmissions, which degrades the performance of APC slightly, as we will see in the next section in detail.

### 7.5.2 Experiments

Fig. 7.11 shows the experimental results for 64 kb/s CBR VoIP traffic. As we can see, regardless of the problem mentioned above, the average uplink and downlink delay as well as 90th percentile ones are well balanced until the number of VoIP calls reaches the capacity, increasing the capacity from 15 calls to 17 calls.

As mentioned in the previous section, there is slight timing gap between the completion of a transmission and setting the new CW value. When the firmware receives an ACK frame, it calls a function to notify the completion of a transmission to the driver, the driver gives a command to change the CW size to the firmware, and the firmware sends the next frame using the new CW size. The gap varies depending on the processing time of kernel and firmware, and the effect also depends on the number of frames on the firmware queue. Thus, to check the difference between the CFT values the driver computed and the values the firmware actually applied at the MAC layer, the CFT values at the driver and the number of packets sent contention free were measured, and the frequency and CDF of the two values were plotted. Fig. 7.12 shows the CFT values. The frequency shows the fraction of such CFT. For example, when the fraction of CFT value 17 is 8%, it means that 8% of the CFT used 17 as the CFT value (17 packets were transmitted contention free). We can see a slight difference between the CFT values at the driver and the actual CFT, which degrades the performance slightly. It was also found that the difference determines the performance;



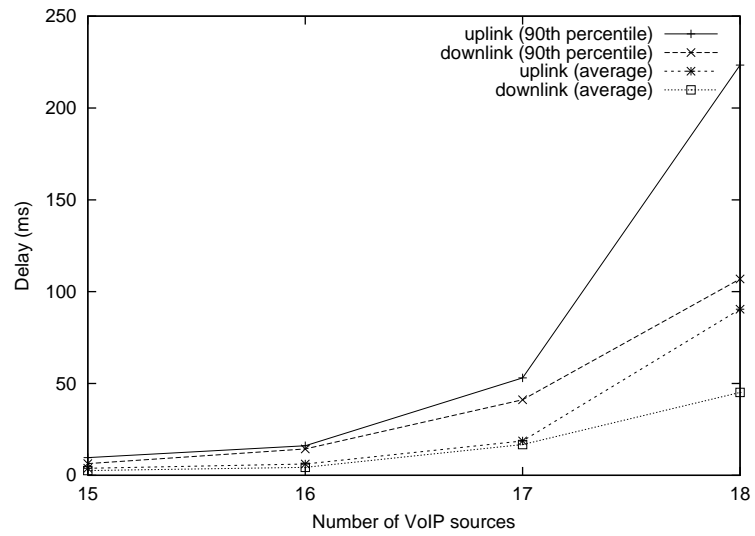


Figure 7.11: Delay as a function of the number of CBR VoIP sources in the experiments

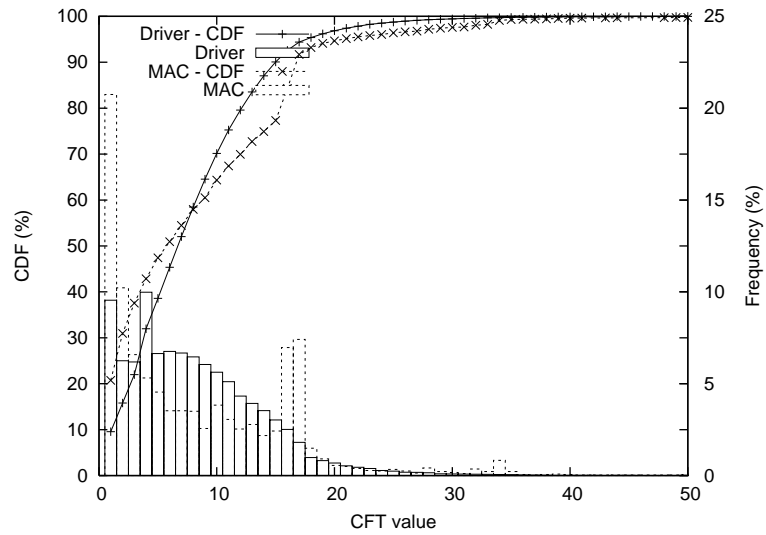


Figure 7.12: CFT values at the driver and actual number of packets sent contention free (the frequency and CDF)

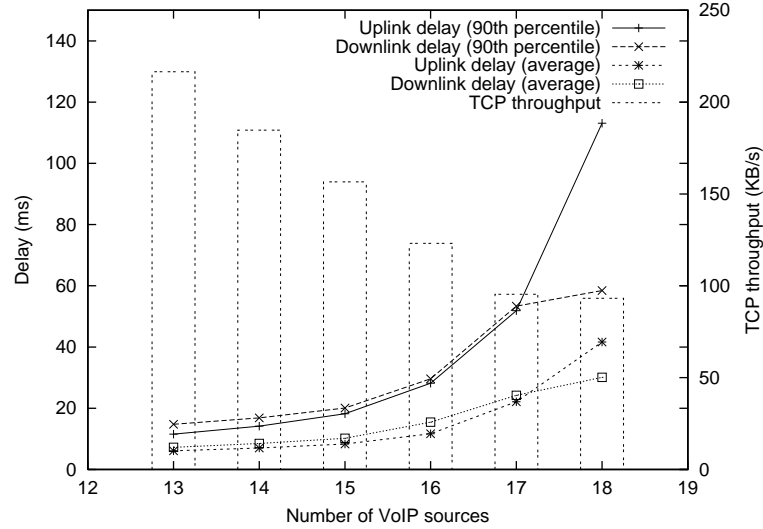


Figure 7.13: Delay and throughput as a function of the number of VoIP sources with TCP traffic in the experiments

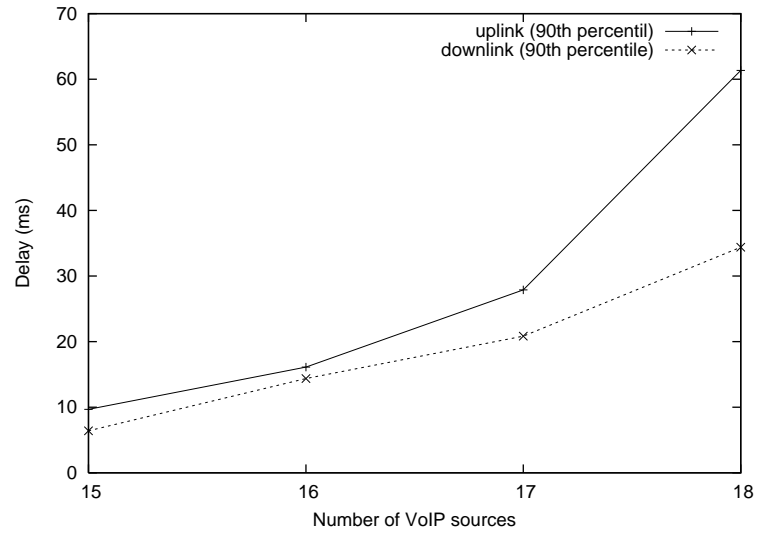
when the difference was small, the two delays were more balanced, and when it was large, the difference of the two delays was also large. Therefore, we can infer that if APC is implemented in the firmware, then the difference would be eliminated, and the APC will balance the two delays more effectively.

Fig. 7.13 shows the experimental results with TCP traffic. In order to give a higher priority to VoIP traffic, TCP traffic used AC\_BK as the 802.11e access category. We can see that the delay of VoIP traffic is not affected by TCP traffic, balancing uplink and downlink delay very well. Comparing with experimental results of 802.11e EDCA in Section 5.6.2 (Fig. 5.20), the capacity of APC is larger than that of EDCA, allowing also higher TCP throughput; the capacity of EDCA with AC\_VO is 15 calls and the TCP throughput with 15 VoIP calls in EDCA is 130 KB/s. Therefore, we can conclude that APC performs much better than EDCA in terms of both VoIP capacity and TCP throughput.

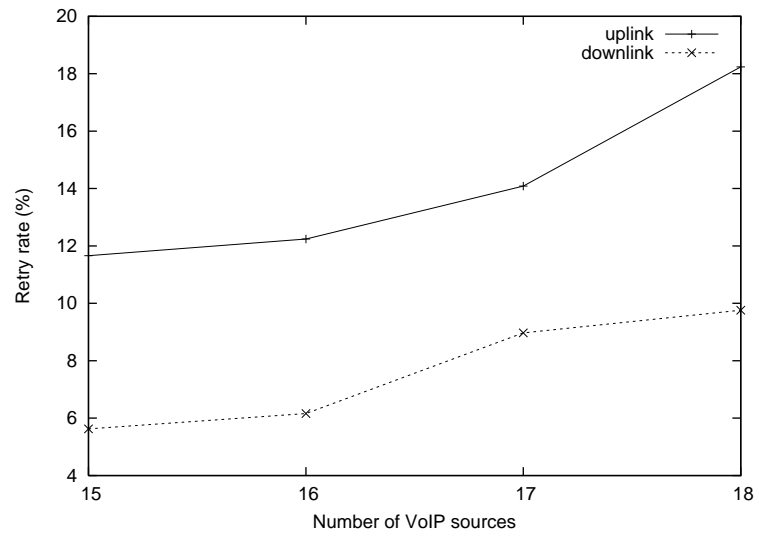
## 7.6 APC and IEEE 802.11e

As explained in Chapter 5, the IEEE 802.11e standard was proposed to protect the QoS of real time services like voice and video, and in the EDCA, which is an extension of DCF, the traffic is differentiated into four access categories, by using different CW, AIFS, and TXOP values. In particular, TXOP works similarly to CFT, and the nodes are allowed to transmit frames without contention for the fixed TXOP duration; for example, in AC\_VO (access category for voice traffic), 3096  $\mu$ s of TXOP duration is granted.

APC works with IEEE 802.11e replacing the TXOP feature at the AP with CFT, and the performance of APC with 802.11e was measured via experiments. Figs. 7.14 show the experimental results. We can see that both uplink and downlink are still balanced until the number of VoIP calls reaches the capacity, and the capacity is also improved to 18 calls. The additional capacity improvement comes from the packet transmission during TXOP at the clients, which eliminates the backoff and reduces the collisions, and thus it also allocates more bandwidth to the AP. Table 7.1 shows the fraction of packets sent using CFT at the AP and during TXOP at clients. We can see that 10% and 17% of the packets are transmitted during TXOP



(a) Delay



(b) Retry rate

Figure 7.14: Delay and retry rate as a function of the number of VoIP sources in IEEE 802.11e

Table 7.1: Packet transmission in APC with 802.11e

MAC protocols	APC	APC+AC_VO	
Number of VoIP sources	17 calls	17 calls	18 calls
Fraction of packets sent using CFT at the AP	37%	41%	48%
Fraction of packets sent during TXOP at clients	0	10%	27%

at clients, with 17 calls and 18 calls, respectively, and thus, the fraction of packets sent using CFT at the AP also increased from 37% to 41% with 17 calls.

Also, we can see from Fig. 7.14(b) that CFT in APC does not increase the retry rate while using TXOP in IEEE 802.11e increase the retry rate significantly as we have seen in Chapter 5. The downlink retry rate is only 14% with 17 VoIP calls, which is less than a half of the downlink retry rate with AC\_VO (30%). This is because the CFT mechanism in APC allows the wireless card firmware of the AP to detect any transmission during CFT, by using 1 slot time + SIFS as its IFS, and thus it prevents collisions when clients transmit frames during CFT of the AP. This is also why the capacity with APC AC\_VO is larger than that of 802.11e AC\_VO.

## 7.7 APC without knowledge of the client queue size

In order to use the ratio of the queue size of the AP and a client as the priority value of the AP, the AP needs to know the queue size of all wireless clients as well as its own queue size. In the simulations and experiments, this was implemented by inserting the queue size of each client to the body frame of each VoIP packet. However, this method requires changes not only in the AP but also in the clients. Even though it can be implemented only with modifying the wireless card drivers of clients, it would make APC more practically deployable if APC requires changes only at the AP. Therefore, this section proposes two approaches to avoid changing clients.

### 7.7.1 Estimating the queue size of the client

One way to implement APC without changing the client is to estimate the queue size of the clients. The queue size of the clients is the number of packets generated in the application layer minus the number of packets sent to the AP. The AP can compute the number of packets sent to itself. Also, it can calculate the number of the packets generated at the wireless clients if it knows the number of active wireless clients, by dividing it by the packetization interval, and the number of active wireless clients can be estimated by checking the received packets from wireless clients; for example, if 10 wireless clients are sending VoIP packets with 0.02 s packetization interval, 500 (=10/0.02) packets are generated from all wireless clients every second. Therefore, the equation to estimate the average queue size of wireless clients can be summarized as follows:

$$Q_i = Q_{i-1} + \frac{\sum_{j=1}^{N_i} (t_s - R_j)}{N_i},$$

where  $Q_i$  is the estimated average queue size of wireless clients at  $i$ th sampling time,  $N_i$  is the number of active wireless clients at  $i$ th sampling time,  $t_s$  is sampling interval in milliseconds,  $R_j$  is the packetization

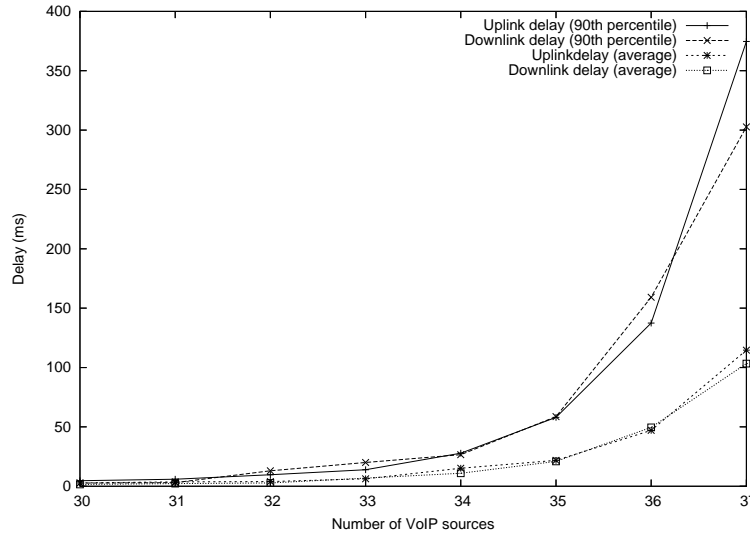


Figure 7.15: Delay as a function of the number of VoIP sources using the estimated queue size of clients

interval of the wireless clients  $j$  in milliseconds, and  $R_j$  is the number of packets the AP received from the wireless client  $j$ .

This algorithm was implemented and integrated into APC. Fig. 7.15 shows the simulation results of APC with the estimated queue size. Comparing this with Fig. 7.4(b), we can see that there is no difference in the performance between two results, thus confirming that the estimated queue size can be used instead of the actual queue size of wireless clients to implement APC.

### 7.7.2 Controlling the downlink delay with PID control

The main goal of the PID control approach is not the fair resource distribution between uplink and downlink, but maximizing the usage of CFT at the AP until the downlink delay meets the QoS requirement. As we have seen, using CFT decreases the waste of bandwidth from backoffs and collisions, and thus the more we use CFT, the more we can increase the utilization. The only problem is that the uplink delay increases because uplink transmissions need to be deferred until the AP's CFT finishes. This is the exactly opposite results of the case in DCF. As in the case of DCF, it is not desirable that only either uplink or downlink delay is very low and the other is very high. Thus, we need to use the CFT only until the downlink delay satisfies the condition for QoS, which is 60 ms as we have seen in Chapter 5. In this case, the downlink delay keeps 60 ms, and the capacity is determined by the uplink delay. Even though uplink delay and downlink delay are not always balanced, at least when the number of VoIP calls reaches the capacity, the downlink and delay will be balanced.

In order to control the downlink delay to 60 ms, PID (Proportional, Integral, and Derivative) control method [16], which is the most general control method, was used.  $P$  reacts to the current error,  $D$  reacts to the change of errors, and  $I$  reflects the summation of the past errors. Generally,  $P$  changes the output proportionally to the current error,  $D$  quickly moves the output to the target, and  $I$  changes the output slowly eliminating the oscillations from other terms [16]. Since the downlink delay is already very sensitive to the transmission rate, we do not use  $I$  term in APC, and because lower delay than the target value does not hurt the system, we use  $P$  term only when the delay exceeds the target delay.  $D$  term is

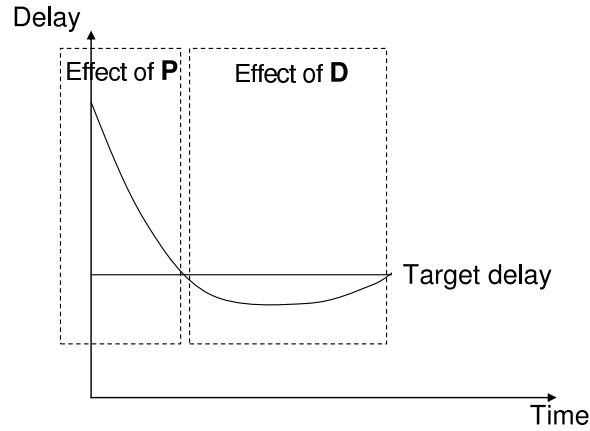


Figure 7.16: Effect of P and D terms on delay

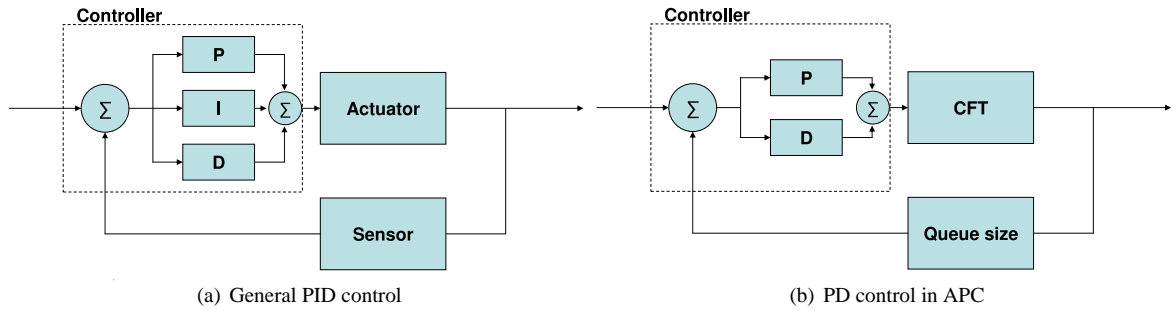


Figure 7.17: PID controller in APC

used to decrease the transmission rate slowly and control it precisely. Fig. 7.16 shows the effect of  $P$  and  $D$  terms.

Figs. 7.17 show the diagram for the general PID controller and the PD controller used in APC. The sensor measures the queue size of the AP, and the queue size is converted to an estimated delay using a QP method, which will be explained in the next chapter. Then, the error between the target delay and the estimated delay is processed in the PID controller, and then the next CFT is computed. Finally, the new CFT value is applied in the CFT (actuator).

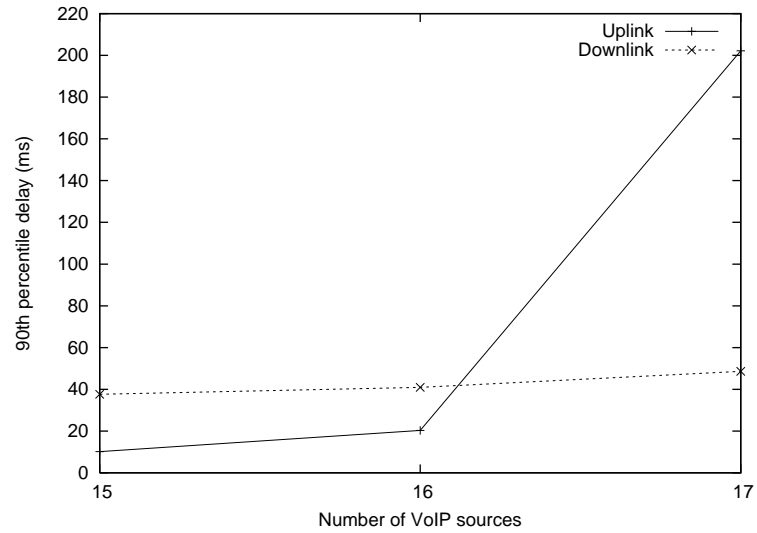
In the controller, CFT values is computed as follows:

$$\text{CFT value} = P + D$$

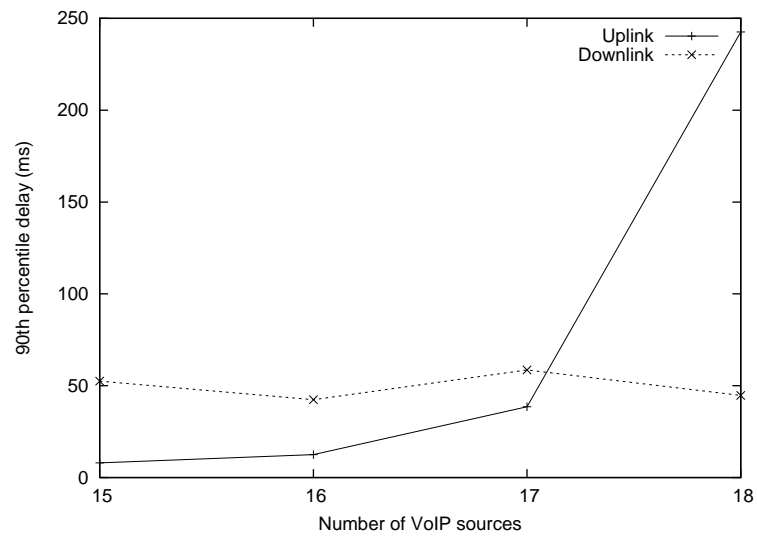
$$P = K_P \times \text{err}, \text{ (when } \text{err} > 0 \text{)}$$

$$D = K_D \times dt(\text{err})$$

,where  $K_P = 1$  and  $K_D = 1$ . By doing this, when the delay exceeds the target delay, CFT values increases by the error according to the  $P$  factor, and thus, the delay drops to the target delay quickly. When the delay decreases below the target delay,  $D$  factor works and the CFT value starts to decrease slowly, preventing oscillations of delay, and then the delay converges to the target delay.



(a) 50 ms of target delay



(b) 60 ms of target delay

Figure 7.18: Delay as a function of the number of VoIP sources of APC with the control method

The approach was implemented in the MadWifi driver, and the performance was measured in the ORBIT testbed. Figs. 7.18 show the experimental results. First, to check the performance of the controller, the target delay was set to 40 ms, and Fig. 7.18(a) shows the uplink and downlink delay. We can see that the downlink delay remains 40 ms even if the number of VoIP sources increases, while the uplink delay significantly increases. The capacity is 16 calls, because the controller allowed too much resources for downlink to keep 40 ms downlink delay. Fig. 7.18(b) shows the delay with the target delay of 60 ms, which maximizes the capacity. The downlink delay keeps 60 ms and the capacity increased to 17 calls due to the fair resource distribution at the time of 17 calls. From the two figures, we can confirm that the controller can adjust the downlink delay very well, and the APC with the control method can allocate the resources efficiently, maximizing the capacity, without the knowledge of the client queue size.

## 7.8 Related work

Many papers have studied fairness among wireless nodes ([11] [3] [1] [61]), and some papers have also considered the fairness between the AP and wireless nodes ([38], [33]). However, they have focused only on throughput fairness and failed to consider the balance of the end-to-end delay, which is more important for VoIP traffic. I have also confirmed via simulations that the Jain's Fairness Index [32], which was computed including all wireless and wired nodes, is very close to 1, meaning that all nodes share the throughput equally, even when uplink and downlink delay are significantly unbalanced.

The following papers considered the balance of uplink and downlink delay. Wang et al. [86] introduced the New Fair MAC to improve the fairness and delay of VoIP traffic. When a station wins a transmission opportunity, it is allowed to transmit a burst of packets instead of a packet. However, allowing stations to transmit a burst of packets does not help much for VoIP traffic because only one VoIP packet is sent every packetization interval. Also, for fairness between stations, they introduce Max Transmission Time (MTT) in the client transmissions, which is similar with TXOP at client side and different from CFT in that the CFT value changes dynamically. Considering the packetization intervals are usually 10 ms to 40 ms and the uplink delay is very low even when the number of VoIP nodes exceeds the capacity, as we have seen, only one packet will be sent during the MTT as in DCF, and for this reason, the uplink delay decreased by only a few milliseconds.

Casetti et al. [8] improved the fairness for VoIP between nodes and the AP in IEEE 802.11e Enhanced Distributed Channel Access (EDCA) by differentiating frames based on traffic type and also direction. They found the optimal Contention Window (CW) values for the best fairness and throughput of VoIP traffic via simulation and improved the capacity of VoIP traffic by around 15%. However, they tested the optimal CW values with only one type of VoIP traffic and failed to show that the optimal value works for other types. We have already seen that the optimal parameters should be changed according to the number of VoIP nodes, type of VoIP traffic, and traffic volume, and also changing CW values to control the priority of frames has limitations, as shown in Section 7.4.2.

## 7.9 Conclusion

As the number of VoIP sources increases, in DCF the downlink delay increases significantly while the uplink delay remains low. This is because every wireless node including the AP has the same chance to transmit frames in DCF, while the AP needs to transmit more packets than wireless nodes. In this chapter,



I have proposed APC, which controls the priority of the AP adaptively according to the traffic volume and balances the uplink and downlink delay, by allowing the AP to transmit  $Q_{AP}/Q_C$  packets contention free.

It was shown that APC balances the uplink and downlink delay, and the APC algorithm was implemented using the QualNet simulator, and it was shown that APC balances the uplink and downlink delay effectively in VoIP traffic with various packetization intervals.

Furthermore, APC was implemented using the MadWifi driver and the performance was evaluated in a wireless test-bed. Also, APC can be combined with 802.11e and it was confirmed via experiments that APC works better than the standard 802.11e in terms of VoIP capacity and throughput of background traffic.

## **Part III**

# **QoS and Call Admission Control**

## Chapter 8

# Call Admission Control using Queue size Prediction by Computing Additional Transmissions (QP-CAT)

### 8.1 Introduction

When the number of VoIP calls in a BSS exceeds the capacity, the QoS of all calls significantly deteriorates, and thus we need to protect the QoS of existing calls using call admission control. Call admission control in IEEE 802.11 differs from that in Ethernet due to the characteristics of 802.11 wireless networks. While most of the admission control algorithms for wired networks are based on the end-to-end QoS, that in wireless networks is mainly to protect the QoS of flows between the AP and clients in a BSS. Therefore, most of the legacy admission control algorithms for wired networks are not applicable. Furthermore, the admission decision in wireless networks is more difficult than that in wired networks. To make admission decisions, the AP needs to predict the QoS of all VoIP flows including new flows. However, the voice quality changes according to the data rate, collision rate, and other factors, even with the same number of active nodes and same VoIP traffic type. Therefore, this chapter introduces a new call admission control, QP-CAT (Queue size Prediction using Computation of Additional Transmission), which uses the number of VoIP packets in the queue of the AP as the metric for the quality and accurately predicts the increase of the queue size of due to the admission of new VoIP flows.

### 8.2 Correlation between the queue size and downlink delay

An accurate metric to estimate the channel condition is the most critical factor for call admission control, and QP-CAT uses as the metric the queue size of the AP, which is easy to compute and allows accurate estimation of the QoS of all VoIP flows. To check the accuracy, the correlation between the queue size of the AP and the downlink delay of VoIP traffic was identified, because downlink delay represents the QoS for all VoIP flows for the following reasons: first, we have seen in Chapter 5 that only the downlink delay increases as the number of VoIP sources increases. This is because of the unfair resource distribution between uplink and downlink explained in Chapter 7. Secondly, the downlink delay exceeds the threshold

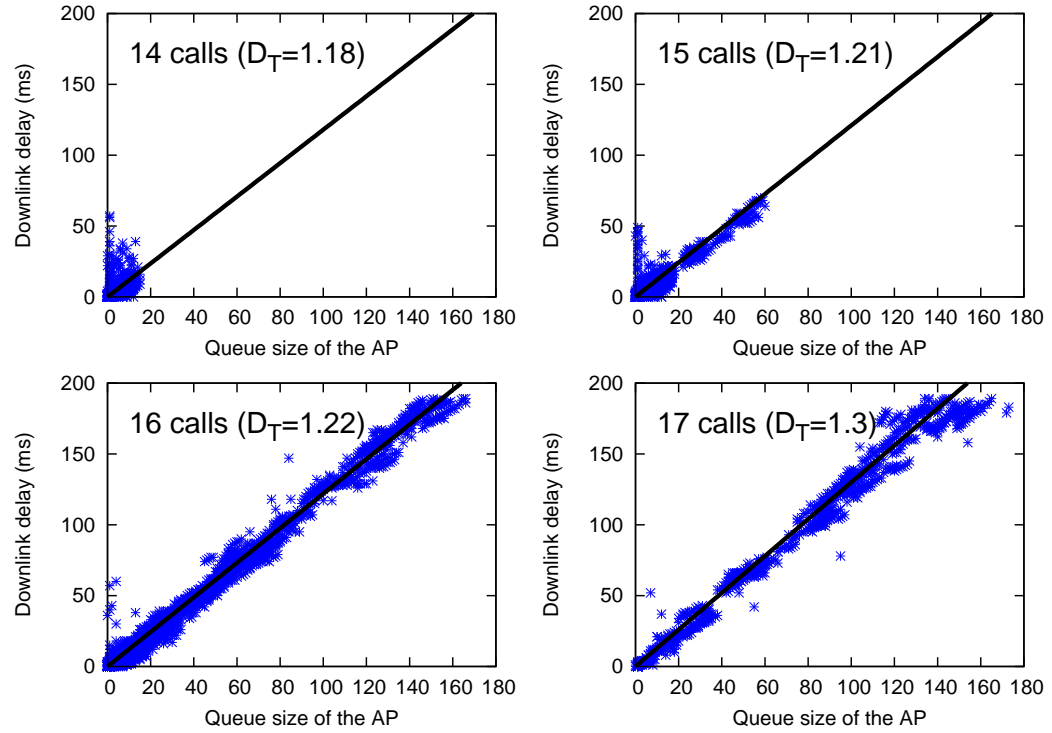


Figure 8.1: Correlation between the queue size of the AP and instant downlink delay in different number of VoIP sources: each point represents the downlink delay of a frame and the queue size of the AP when the frame was queued, and the straight line is the theoretical model.

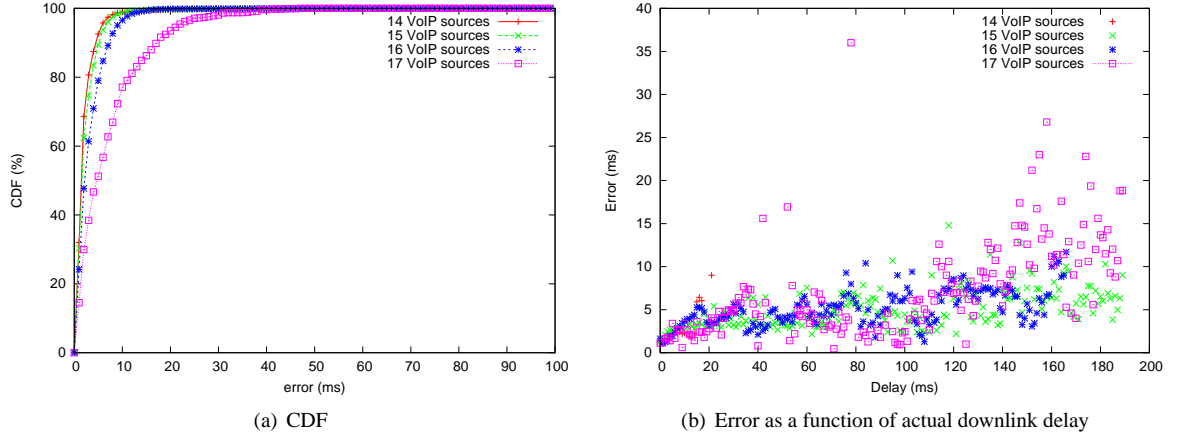


Figure 8.2: The errors between the actual downlink delay and the estimated delay using the queue size of the AP

first before the packet loss rate does, when the number of VoIP sources exceeds the capacity [73], as shown in Section 5.5.8. Lastly, all VoIP sources in a BSS have the similar downlink delay, because the queuing delay at the AP dominates the downlink delay.

In order to identify the correlation between downlink and the queue size of the AP, the queue size of the AP and downlink delay were measured in the ORBIT wireless test-bed, described in Section 5.4. G.711 as voice codec with 20 ms packetization interval yielding 64 kb/s CBR VoIP flows and 11 Mb/s fixed transmission rate were used in 802.11b. Also, 14 to 17 VoIP sources were used in the measurement because the channel starts to get congested at 14 or more VoIP sources using the configurations. Fig. 8.1 shows the experimental results<sup>1</sup>. We can see that the queue size of the AP and downlink delay strongly correlate with each other; as the queue size at the AP increases, the downlink delay also increases linearly.

I have verified the correlation also by analysis. The downlink delay ( $D$ ) is composed of the queuing delay ( $D_Q$ ), transmission delay ( $D_T$ ), and propagation delay. We can ignore the propagation delay because it is very small. Then,  $D = D_Q + D_T$ . We can compute the queuing delay ( $D_Q$ ) by multiplying the transmission delay ( $D_T$ ) to the queue size ( $Q$ ) from Little's law ( $D_{system} = Q_{system} / \mu_{system}$ ). Then, we can compute the queuing delay of the AP ( $D_Q$ ) as follows:

$$D_Q = Q \cdot \frac{1}{\mu_{AP}} = Q \cdot D_T$$

Therefore, the downlink delay ( $D$ ) becomes

$$D = Q \cdot D_T + D_T = (Q + 1) \cdot D_T,$$

where  $D_T$  is the average time to transmit one VoIP packet including all overhead.  $D_T$  was measured at the driver in the experiments, as shown in Fig. 8.1. In the figure, the straight line is the theoretical relationship between the queue size of the AP and downlink delay of the VoIP traffic, and we can see that the experimental results are exactly following the theoretical model.

<sup>1</sup>The maximum queue size in the MadWifi driver is 50 frames by default. I extended the maximum queue size to 200 frames only in the experiments to see the correlation more clearly.

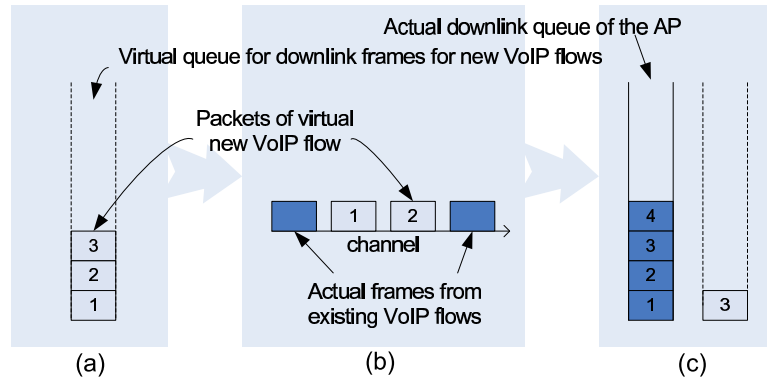


Figure 8.3: Basic concept of QP-CAT

In order to identify the accuracy of the model, the cumulative distribution function (CDF) of the errors between the actual downlink delay and the estimated one using the theoretical model was computed and plotted in Fig. 8.2(a). We can see that the 95th percentile error is below 10 ms with 14 to 16 VoIP calls. With 17 VoIP sources, the error becomes larger because the channel is extremely overloaded and the transmission delay fluctuates significantly due to the increased retransmissions and deferrals. However, this is not a problem because when the call admission control is applied appropriately, this situation never happens. Fig. 8.2(b) shows the accuracy of the model as a function of the downlink delay. It shows that with 14 to 16 VoIP sources, the model maintains a similar accuracy even when downlink delay increases.

As shown in Figs. 8.1 and 8.2, the downlink delay can be accurately estimated using the queue size of the AP and the theoretical model, and thus the queue size of the AP is an accurate metric for the call admission control.

### 8.3 Queue size Prediction (QP) using Computation of Additional Transmission (CAT)

The best way to decide the admission of a new VoIP flow is to measure the queue size of the AP after it has been admitted. However, it is not appropriate to disconnect the admitted flow when it was discovered that it degrades the QoS of all VoIP flows. Another way is that as in the call admission control methods for wired networks, probing flows can be transmitted instead of actual VoIP flows [49], but this wastes a certain amount of bandwidth because clients should keep probing in wireless networks, which is a critical disadvantage because of the limited bandwidth.

QP-CAT is a simple and accurate technique to predict the queue size, without wasting any bandwidth, but achieving the same performance as using actual probing. The basic concept of the QP (Queue size Prediction) is to predict the future queue size of the AP using the simulation of a new VoIP flow and the Computation of Additional Transmission (CAT), where the number of packets to be transmitted additionally is computed in real time by monitoring the channel status. The basic operation of QP-CAT is shown in Fig. 8.3. First, VoIP packets from a new virtual VoIP flow is generated following the behavior of the traffic and inserted into a virtual queue (Fig. 8.3 (a)). Then, by monitoring the channel, we compute how many VoIP frames can be additionally transmitted using CAT (Fig. 8.3 (b)). Lastly, the number of additionally transmittable frames is subtracted from the virtual queue size, and then the total number of

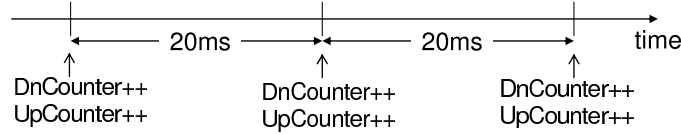


Figure 8.4: Emulation of a new VoIP flow with 20 ms packetization interval

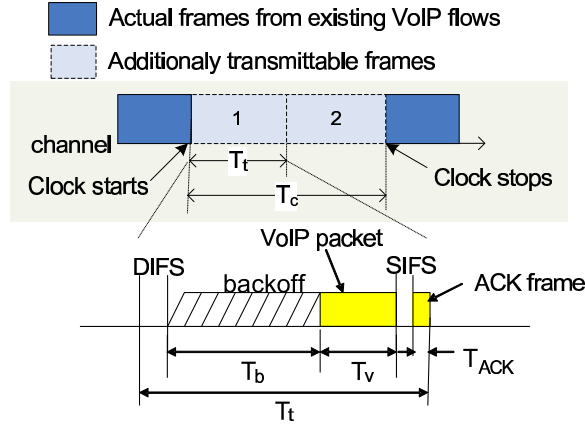


Figure 8.5: Computing the number of additionally transmittable VoIP packets

packets in the actual queue and virtual queue becomes the predicted queue size.

### 8.3.1 Emulation of new VoIP flows

In order to emulate a new VoIP flow, two counters, *UpCounter* and *DnCounter*, which count the number of the uplink and downlink packets of a new VoIP flow, respectively, are introduced. The counters are incremented following the behavior of the new VoIP flow. For example, for the VoIP traffic with 20 ms packetization interval, both of the counters are incremented by one every 20 ms (Fig. 8.4). The counters are decremented in real time according to the number of packets computed using CAT. They are decremented alternatively because the chance to transmit packets is the same between the uplink and downlink. Consequently, the actual queue size of the AP plus *DnCounter* becomes the predicted future queue size of the case when the VoIP flow is admitted.

### 8.3.2 Computation of Additional Transmission (CAT)

The number of additionally transmittable packets ( $n_p$ ) is computed by looking at the current packet transmission behavior. That is, a clock starts when medium becomes idle and stops when busy medium is detected (Fig. 8.5). When the clock stops,  $n_p$  is computed by dividing the clock time ( $T_c$ ) by the total transmission time ( $T_t$ ) of a VoIP packet (Eqn. 8.1).

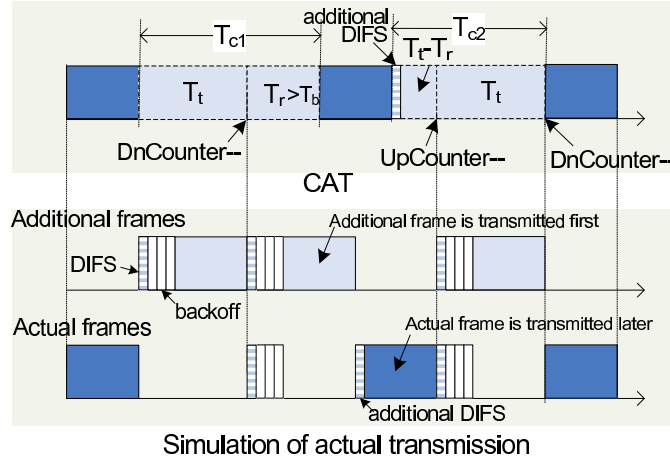
$$n_p = \lfloor T_c / T_t \rfloor \quad (8.1)$$

The transmission time of a VoIP packet comprises all headers in each layer, IFSs, backoff and an ACK frame (Fig. 8.5). Thus, the transmission time ( $T_t$ ) is computed as follows:

$$T_t = T_{DIFS} + T_b + T_v + T_{SIFS} + T_{ACK},$$

Table 8.1: IEEE 802.11b parameters (11 Mb/s)

Parameters	Time ( $\mu s$ )	Size (bytes)
PLCP Preamble	72.00	18
PLCP Header	24.00	6
MAC Header+CRC	24.73	34
SIFS	10.00	
DIFS	50.00	
Slot	20.00	
$CW_{MIN}$	31 slots	

Figure 8.6: Handling the remaining time ( $T_r$ ): when  $T_r > T_b$ 

where  $T_v$  and  $T_{ACK}$  are the time for sending a voice packet and an ACK frame, respectively,  $T_b$  is the back-off time,  $T_{DIFS}$  and  $T_{SIFS}$  are the durations of DIFS and SIFS. The backoff time is  $Number\ of\ Backoff\ Slots \times T_{slot}$  where  $T_{slot}$  is a slot time, and  $Number\ of\ Backoff\ Slots$  has a uniform distribution over  $(0, CW_{MIN})$  with an average of  $(T_{slot} \times CW_{MIN}/2)$  (Fig. 8.5).

For example, with 64 kb/s VoIP traffic with 20 ms packetization interval, the voice data size is 160 B, the VoIP packet size including IP, UDP and RTP [70] headers becomes 200 B, and then the total transmission time becomes 791.82  $\mu s$  including the average backoff time (310  $\mu s$ )<sup>2</sup> and 14 B ACK frame (130.18  $\mu s$ ), in IEEE 802.11b with 11 Mb/s transmission rate (refer to Table 8.1 for the 802.11b parameters). Thus, for example, when  $T_c$  is 1200  $\mu s$ , then  $n_p$  is 1 according to Eqn. 8.1.

However, in real environments, the frames from new VoIP sources are not always transmitted in between the frames from existing sources. Sometimes, they collide with the frames from existing VoIP sources, and the existing VoIP sources need to defer their transmissions due to the new flows. Therefore, the effect of collisions and deferrals need to be also considered in CAT, by handling the remaining clock time ( $T_r$ ) as described below.



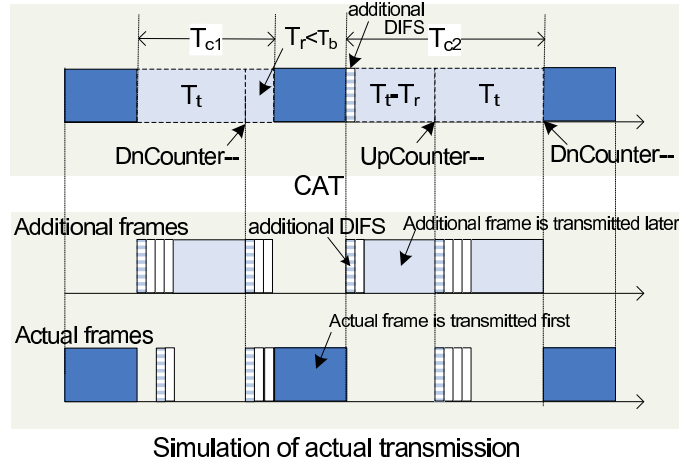


Figure 8.7: Handling the remaining time ( $T_r$ ): when  $T_r \leq T_b$

### Handling the remaining clock time ( $T_r$ )

Generally,  $T_c$  is not always a multiple of  $T_t$ , and we have remaining time ( $T_r$ ).  $T_r$  is defined as follows:

$$T_r = T_c - n_p \times T_t \quad (8.2)$$

When  $T_r > 0$ ,  $T_r$  is added to the next idle time ( $T_{c2}$ ), causing an additional DIFS. That is,  $T_c = T_{c2} + T_r - T_{DIFS}$ . We can check the reason in two possible cases,  $T_r > T_b$  and  $T_r \leq T_b$ . First, when the remaining time is larger than the backoff time ( $T_r > T_b$ ), the virtual additional frame needs to be transmitted first, and then the actual frame would be transmitted, if the new VoIP call is admitted. In this case, the additional frame interrupts the backoff time of the actual frame from an existing flow, and another DIFS is required to resume the backoff and transmission according to the 802.11 standard. This is why additional  $T_{DIFS}$  is added in the next computation of  $n_p$ . Fig. 8.6 shows the emulation result and comparison with the result of CAT. We can see that the total number of transmitted frames during  $T_{C1}$  and  $T_{C2}$  in CAT is the same as the emulation result (three additional frames). Secondly, when the remaining time is larger than or equal to the backoff time ( $T_r \leq T_b$ ), the actual frame is transmitted first, and then the additional frame can be transmitted (Fig. 8.7). This case corresponds to the real transmission, and we can see that CAT is consistent with the simulation result.

### Emulation of collisions

To predict the queue size at the AP more accurately, we need to consider collisions, which congest the channel and cause additional delay. A simple way to emulate collisions is to apply the average retry rate for a certain amount of time to additional transmissions, since the retry rate can be easily measured in the firmware or the driver. For example, if the average retry rate of downlink traffic is 3%, then three is added to *DnCounter* every time the counter is incremented by 100. However, while this method is easy to implement, it cannot reflect the increase of collisions due to the admission of new calls. Therefore, in CAT we emulate collisions following the actual collision mechanism.

<sup>2</sup>Even though the average backoff time was used in the example, the backoff time is generated using a random value between 0 and CW size in the implementation, as in the standard.

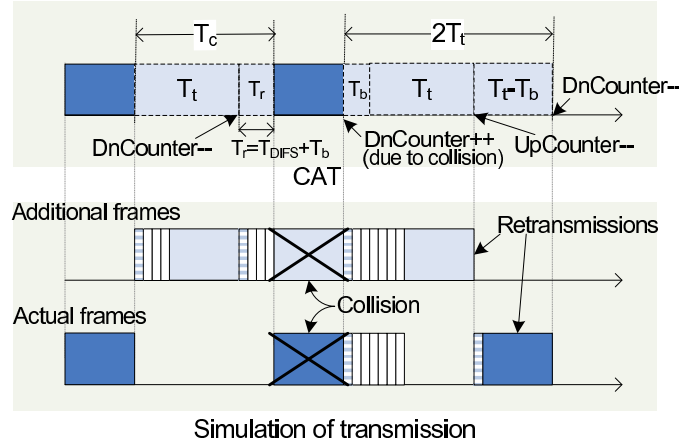


Figure 8.8: Emulation of collisions: during  $2T_t$ , only one additional frame can be transmitted due to the collision, in the end

As we can see in Fig. 8.8, if  $T_r$  is exactly the same as the backoff time plus DIFS (that is,  $T_r = T_b + T_{DIFS}$ ), transmission of the additional frame is attempted at the same time as transmission of the actual frame (refer to the emulation part in Fig. 8.8). Thus, it is considered as a collision in CAT. If an actual collision happens, both frames need to be retransmitted (we ignore the capture effect, where a packet can be transmitted through the collision, because it rarely happens). However, the actual frame is not retransmitted here because collision did not happen in the real transmissions. Therefore, the actual frame retransmission needs to be emulated by adding a virtual frame to *DnCounter* (since the downlink frame transmission is delayed due to the retransmission, the effect is the same). That is, the impact of a collision is transmission of additional two frames, and thus  $2T_t$  is considered as one frame transmission in CAT, eventually.

In summary, the algorithm of QP-CAT is shown in Fig. 8.9. First, we measure  $T_c$  by checking the channel idle and busy time. Next, we check if there is any remaining time ( $T_r$ ) from the previous computation. If any, check if the virtual collision happened and handle the collision. Otherwise, compute new  $T_c$  using  $T_r$ . Then, we compute the number of additionally transmittable frames ( $n_p$ ), update *DnCounter* using  $n_p$ , and compute the future queue size ( $Q_p$ ) using the actual queue size of the AP ( $Q_A$ ) and *DnCounter*.

## 8.4 Simulations and their results

In order to verify the accuracy of the QP-CAT algorithm, QP-CAT was implemented in the same QualNet 3.9 simulator [64] as the other simulations shown in earlier chapters.

### 8.4.1 Simulation setup

The same simulation setup was used as other simulations, explained in Chapter 5. Voice packets were transported using UDP and RTP, and various VoIP packet size and packetization intervals were used to evaluate the performance of QP-CAT extensively. Also, IEEE 802.11b and a fixed 11 Mb/s data rate were used. RTS/CTS was not used because it is not generally used in VoIP traffic due to its overhead.

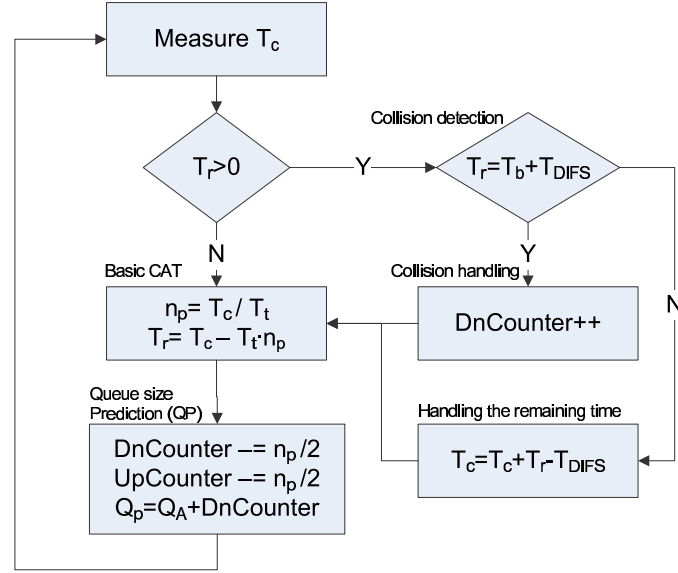


Figure 8.9: Flowchart of QP-CAT algorithm

### 8.4.2 Simulation results

Fig. 8.10 shows simulation results with 16 to 18 CBR VoIP calls (32 kb/s and 20 ms packetization interval). Fig. 8.10(a) shows the actual queue size of the AP with 16 and 17 VoIP calls and the predicted queue size for an additional call; during the first 70 seconds, there exist 16 actual calls, and the predicted queue size of 17 calls case indicates that an additional call does not deteriorate QoS of existing calls. Hence, the actual 17th call is initiated and admitted at 70 seconds. After the admission of 17th call, we can see that the queue size remains stable as QP-CAT predicted. Also, QP-CAT predicts that the queue size would increase to 400 packets causing buffer overflow if the 18th call is admitted. Fig. 8.10(b) shows the case of 18 actual calls compared with the case of 17 calls plus one virtual call. At the time of 70 seconds, 18th call starts and the actual queue size exceeds 400 packets causing buffer overflow, as predicted by QP-CAT.

Like all admission control algorithms that monitor the channel status to decide admissions, such as [17] and [93], the prediction takes time in QP-CAT. The longer we monitor the channel, the better decision we can make. In Fig. 8.10, it took five seconds for the predicted queue size to increase up to 400 packets. According to experiments in this study, the convergence time depends on the channel status. If the channel is almost saturated with existing calls, the predicted queue size increases very fast, and if an additional call exceeds the capacity very slightly, then the queue size fluctuates and increases very slowly. According to simulations, in the worst case, it can take up to 20 seconds for the queue size to converge to the maximum buffer size. However, note that QP-CAT does not cause a 5 to 20 second call setup delay since the prediction by QP-CAT can be done continuously, ahead of the actual call arrival, without any overhead of probing traffic. Thus, if the call arrival rate is larger than 20 seconds, it is not a problem at all, but if more than two calls arrive at the same time or within a short time, the second call user needs to wait until the decision is made via additional measurement after the first call is added. However, in QP-CAT, this problem can be avoided by serial execution of QP-CAT, which will be explained in Section 8.6.2.

More simulations were performed with various types of VoIP traffic and different number of VoIP sources. As we can see in Fig. 8.11, CAT can predict the increase of the queue size of the AP in all the

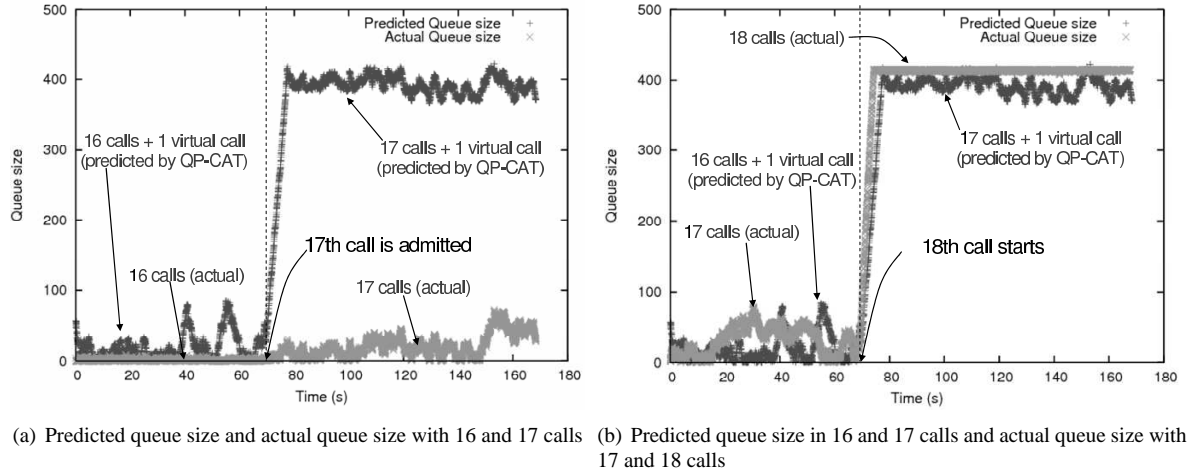


Figure 8.10: Simulation results of QP-CAT with 32 kb/s VoIP traffic using 20 ms packetization interval

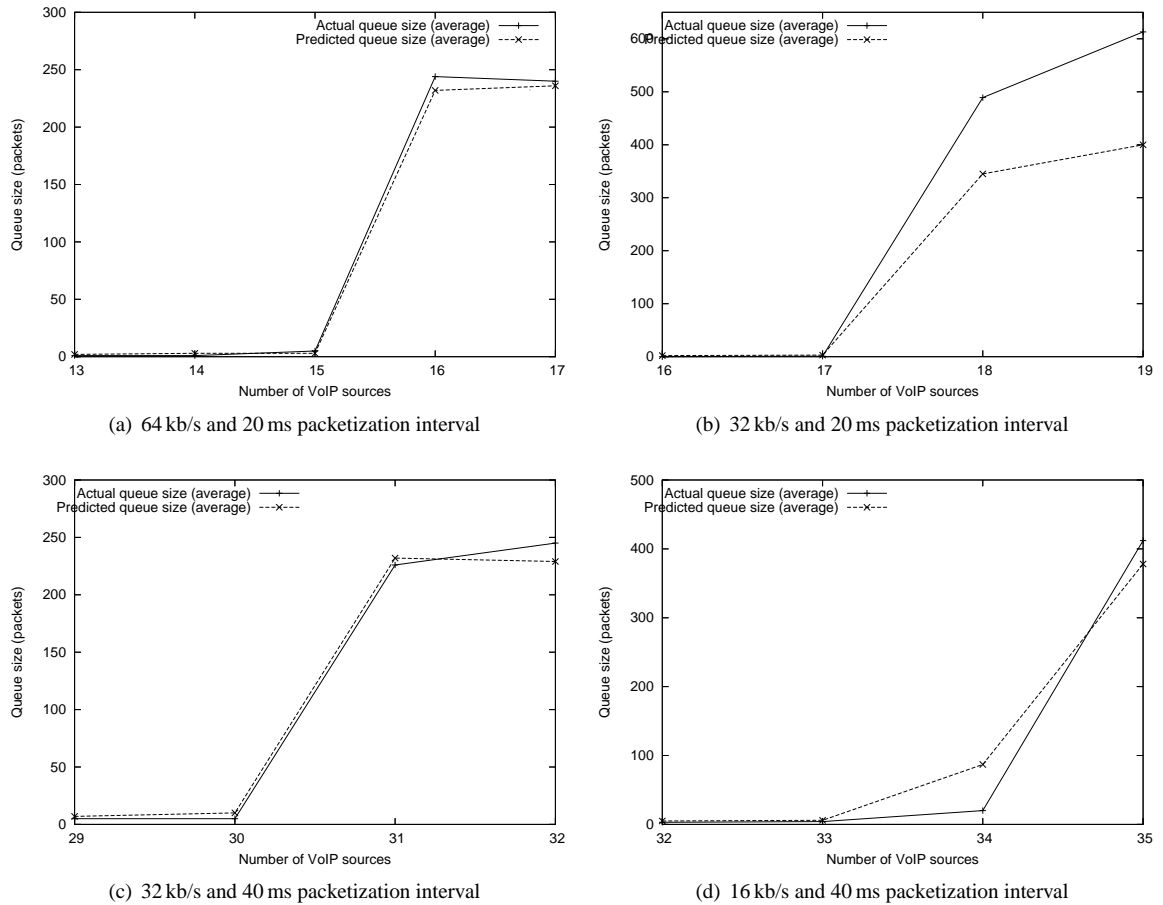


Figure 8.11: Simulation results of QP-CAT with various types of CBR VoIP traffic

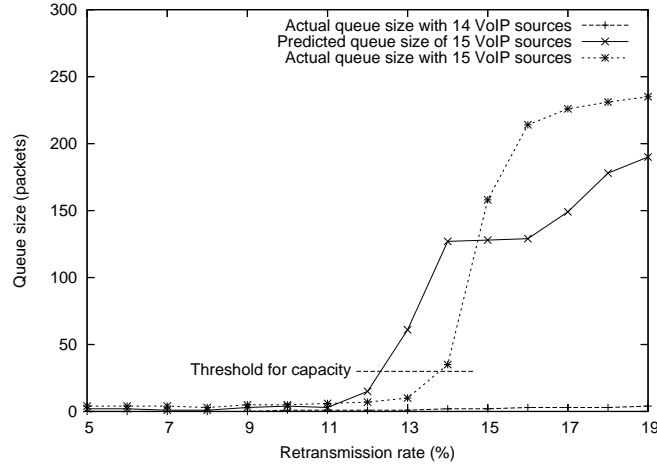


Figure 8.12: Simulation results of QP-CAT with 14 CBR VoIP sources with increasing the collision rate

types of VoIP traffic when the number of VoIP sources exceeds the capacity of each VoIP traffic type.

However, the above four cases could be simple cases where the delay is very small when the number of VoIP sources is below the capacity and an additional VoIP source causes a significant increase of delay. In order to check the performance of QP-CAT more extensively, additional simulations were executed with 14 CBR VoIP sources (64 kb/s and 20 ms packetization interval), increasing the collision rate. Fig. 8.12 shows the simulation results. We can see that when the retry rate increases to more than 13%, the queue size exceeds the threshold value for the capacity (the straight line), and CAT predicts the increase very well. QP-CAT slightly over-predicts at the retry rate of 13%, but it is a false positive case, which does not hurt the QoS of existing flows. When the retry rate increases above 15%, the queue size is underestimated a little bit, but the predicted queue size already exceeded the threshold for QoS, and it is enough to decide to reject the further calls. Note that Fig. 8.12 is not meant to explain the relationship between retry rate and the predicted queue size, but to show that admission control using QP-CAT can protect the QoS of existing calls very well, catching the subtle change of channel status and fully utilizing the channel.

Additionally, the accuracy of QP-CAT was evaluated for VBR VoIP traffic. The available bandwidth is more difficult to measure and the QoS is more difficult to predict for VBR VoIP traffic since the bandwidth used by existing VoIP flows fluctuates much more. For VBR traffic, the usual conversational speech model in ITU-T P.59 [30] was used, as explained in Section 5.2.2, which has 1.0 second average talk-spurts and 1.5 second pauses, resulting in an activity ratio of 0.39. Fig. 8.13 shows the experimental results with 64 kb/s VBR VoIP traffic using 20 ms packetization interval. We can see that QP-CAT maintains roughly the same accuracy as in the CBR case.

## 8.5 Implementation and experiments

In order to verify that QP-CAT is easily implemented in commercial wireless cards and to check the performance in a wireless test-bed, QP-CAT was implemented using the MadWifi wireless card driver and the performance was tested in the ORBIT wireless test-bed.

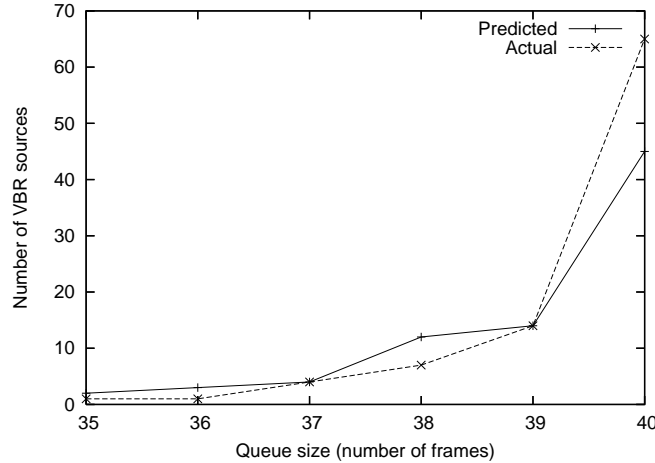


Figure 8.13: Simulation results of QP-CAT with 64 kb/s VBR VoIP traffic using 20 ms packetization interval

### 8.5.1 Implementation

In order to implement the algorithm, we need to know when the channel becomes busy and idle. QP-CAT could be easily implemented in commercial wireless cards because the firmware in wireless cards always check the medium status as part of the CSMA/CA MAC layer. However, I implemented QP-CAT in the MadWifi wireless card driver because the firmware source code of wireless cards is not open to experimenters.

#### Use of the second wireless card

I had to use additional wireless card for QP-CAT due to the limitation of the Atheros chipset; the Atheros chipset notifies the timestamp whenever it finishes receiving (RX timestamp) and transmitting (TX timestamp) a frame, but while the resolution of the RX timestamps is one microsecond, the resolution of the TX timestamps is one millisecond, which is not enough to compute the precise channel idle and busy time for QP-CAT. For this reason, the second wireless card was used at the AP, configuring it as a monitor mode<sup>3</sup> so that it can capture both downlink and uplink frames, and the precise channel time can be computed via the RX timestamps.

#### Computation of the idle time ( $T_c$ )

The idle time ( $T_c$ ) was computed by deducting the transmission complete time of a frame (RX timestamp) from the transmission start time of the next frame, which can be computed by deducting the transmission time of the frame from the RX timestamp. The transmission time of each frame ( $T_t$ ) was computed using the frame size and the data rate used, which can be obtained from the RX descriptor reported from the firmware. Namely,  $T_t = \text{PLCP}^4 + (\text{MAC header} + \text{frame size}) / \text{data rate}$ .

<sup>3</sup>The MadWifi driver supports Virtual AP (VAP) monitor mode, which allows us to monitor and transmit frames at the same time. But, I did not use it because it is known that frames are lost during monitoring and delayed in transmission.

<sup>4</sup>There are two types of PLCP, long and short in 802.11b. The PLCP time is the same for any frame once the type is fixed. Also, 802.11b and 802.11g use different size, but b/g can be determined from the data rate since only 802.11g uses higher rate than 11 Mb/s.

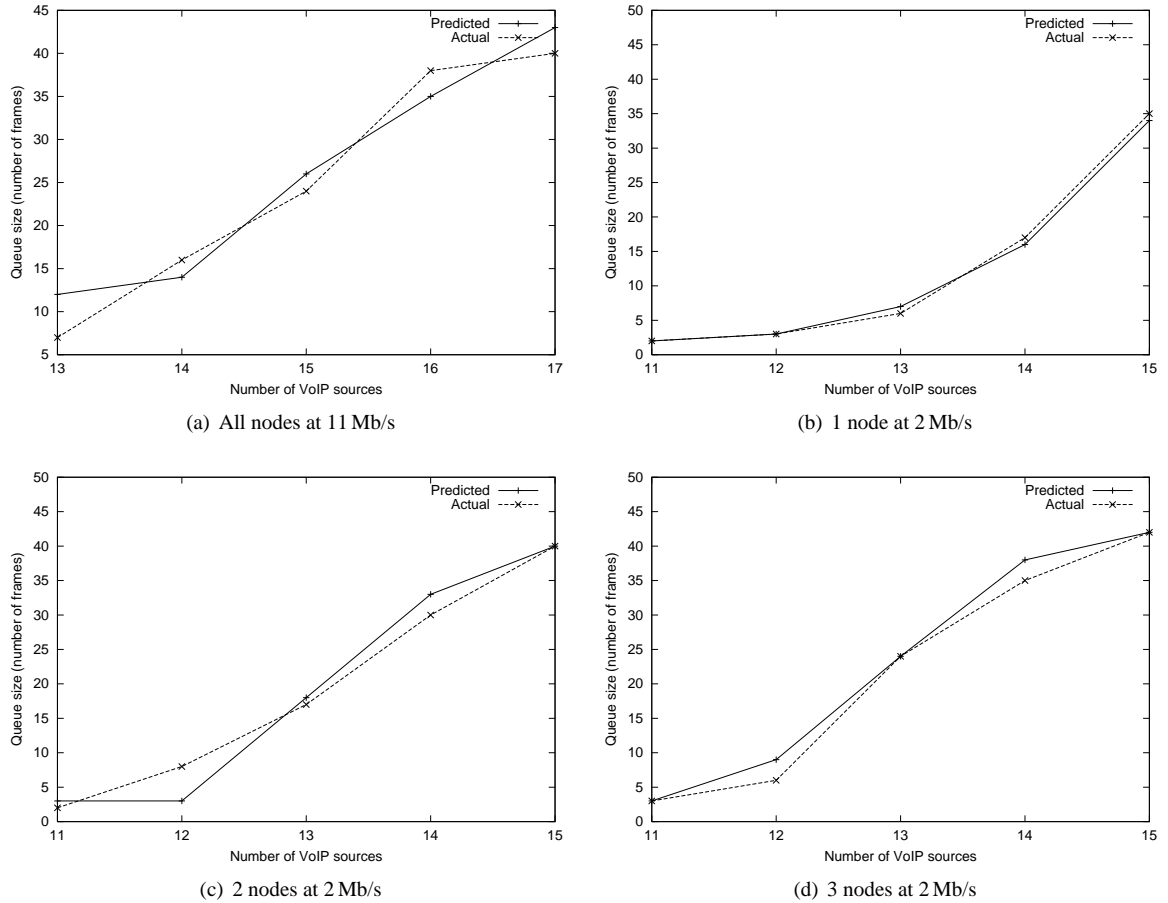


Figure 8.14: Experimental results of QP-CAT with 64 kb/s and 20 ms packetization interval CBR VoIP traffic in various channel status

## 8.5.2 Experimental setup

I have performed experiments in the ORBIT (Open Access Research Testbed for Next-Generation Wireless Networks) test-bed, which is explained in Section 5.4 in detail. The network topology and parameters are the same as those of the simulations in Section 8.4.

## 8.5.3 Experimental results

Figs. 8.14 show the experimental results for QP-CAT using 64 kb/s CBR VoIP traffic with a 20 ms packetization interval. Fig. 8.14(a) shows the actual queue size and the predicted queue size of the AP using QP-CAT, using 11 Mb/s data rate for all nodes. Furthermore, the data rate of one to three nodes was decreased to 2 Mb/s to check the performance of QP-CAT in various channel conditions. We can see that in all four cases, QP-CAT can predict the queue size of the AP very well.

Fig. 8.15 shows the experimental results of QP-CAT for 32 kb/s 40 ms packetization CBR VoIP traffic, where the impact of an additional VoIP source on the channel is smaller and the prediction is more

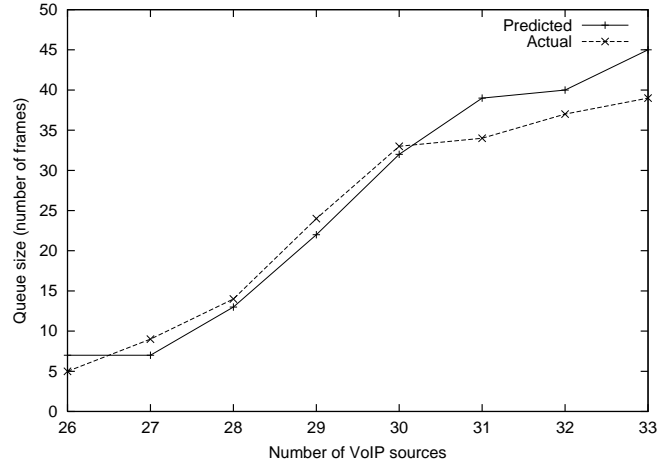


Figure 8.15: Experimental results of QP-CAT for 32 kb/s and 40 ms packetization interval CBR VoIP traffic; the capacity for the VoIP traffic was 28 calls

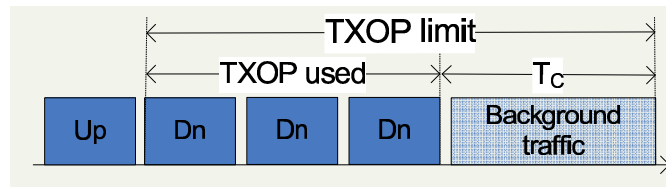


Figure 8.16: QP-CATe: when background traffic is transmitted before using up the TXOP of the AP, the remaining TXOP duration is considered as  $T_C$

difficult. We can see that QP-CAT still works very well for this type of VoIP traffic and predicts the queue size accurately.

We notice that the queue size in experiments is much smaller than that observed in simulations. The difference is explained by the network buffer size (maximum queue size) of the AP in the MadWifi driver and the QualNet simulator. While a buffer size of 50 KB is used by default in the simulator, the MadWifi driver (version 0.9.3) limits the maximum queue size to 50 packets by default. The effect of buffer size on the delay and packet loss was already explained in Section 5.5.8.

## 8.6 Extensions for QP-CAT

### 8.6.1 QP-CAT with IEEE 802.11e (QP-CATe)

In real environments, background traffic such as HTTP and P2P traffic often coexists with VoIP traffic, taking some amount of bandwidth. Thus, the IEEE 802.11 standard committee has proposed IEEE 802.11e to protect the QoS of real time services in WLANs, as explained in Section 1.3.5. Therefore, we assume that 802.11e feature is used when background traffic is allowed to transmit together with VoIP traffic in a BSS because call admission control for VoIP traffic is meaningless without the 802.11e feature; background traffic will degrade the QoS of VoIP traffic, without 802.11e.



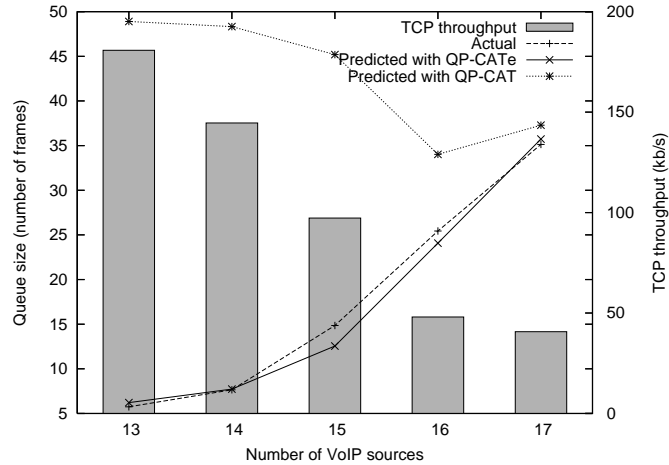


Figure 8.17: Experimental results of QP-CATe with 64 kb/s and 20 ms packetization interval CBR VoIP traffic and a TCP flow; the capacity for the VoIP traffic was 15 calls

In order to support IEEE 802.11e feature, QP-CAT needs slight modification, which I call QP-CATe. When TCP traffic exists, it uses up all available bandwidth, and thus the channel idle time becomes very short even with a small number of VoIP flows. However, when new VoIP flows are added, they have higher priority than TCP traffic, TCP bandwidth is reduced, and the QoS for VoIP is protected as long as the number of VoIP flows does not exceed the capacity. Therefore, QP-CAT needs to be modified to take the priority mechanism into account.

Even though three parameters (CW, AIFS, and TXOP) are used to differentiate traffic in 802.11e, it was explained in Section 5.6.3 that TXOP is sufficient to differentiate the traffic, and thus QP-CATe considers only TXOP to simplify the algorithm. Therefore, when measuring  $T_c$  in CATe, if any background traffic frame is found right after VoIP downlink frames and the TXOP is not finished (that is, the TXOP used is smaller than the TXOP limit), then the remaining amount of TXOP is considered as extra  $T_c$  (Fig. 8.16), because if the AP had more downlink packets due to admission of new flows, TXOP would be fully utilized by transmitting the additional frames.

Additional experiments were performed with TCP background traffic by setting its access category to AC\_BE (access category for best effort traffic) and VoIP traffic as AC\_VO (access category for voice). Fig. 8.17 shows the experimental results. We can see that while QP-CAT over-predicts the queue size due to the TCP traffic, QP-CATe can accurately predict the queue size by considering the effect of TXOP. QP-CATe can be automatically turned on because the wireless card driver or firmware is aware of using 802.11e in the BSS.

QP-CATe is another strong point, which cannot be achieved in other methods that use channel idle time to compute the available bandwidth, like [17] and [93]. They cannot simply ignore the TCP traffic in computing the channel idle time because TCP traffic still takes some amount of bandwidth regardless of the 802.11e priority feature as we can see in Fig. 8.17, decreasing the capacity for VoIP traffic. Therefore, the 802.11e mechanism needs to be considered for accurate admission decision like in QP-CATe.

## 8.6.2 Running multiple instances of QP-CAT

More than one QP-CAT process can be executed at the AP to check multiple types or multiple VoIP calls.

### Parallel execution

Multiple types of VoIP traffic can be checked for admission at the same time by running multiple QP-CAT processes in parallel and independently, which allows the AP to make the admission decision without additional QP-CAT processes when admission is requested for any type of VoIP traffic.

### Serial execution

Serial execution of QP-CAT allows the AP to check the admission of more than one VoIP call. If admissions for two VoIP calls are requested at the same time and one VoIP call was determined to be allowed, then one call can be admitted, but the admission for the other call needs to be investigated after the admission of the first call, which takes time and the user needs to wait until the decision is made.

In QP-CAT, to avoid this problem, the AP can run two QP-CATs back-to-back and the admission for the second new call can be checked at the same time. For the serial execution of QP-CAT, another two counters (*DnCounter2* and *UpCounter2*) need to be added, and they are incremented emulating the behavior of any desired second VoIP traffic. In CAT, if  $n_p$  is larger than the sum of the first two packet counters (*DnCounter1* and *UpCounter1*), then the remaining  $n_p$  value ( $n_p - \text{DnCounter1} - \text{UpCounter1}$ ) is deducted from the second counters, and the queue size of the AP for the two new calls can be predicted by adding *DnCounter2* to the predicted queue size for the first new call ( $Q_p$ ).

## 8.7 Related work

Yang Xiao et al. [90] proposed an admission control algorithm based on the 802.11e EDCA MAC layer. The AP computes the available bandwidth using TXOP duration of current flow and announces it to clients. While this method guarantees a certain amount of bandwidth, it does not guarantee low delay. Because of it, this approach is mainly applicable to video traffic.

Pong et al. [63] estimate the available bandwidth with an analytical model. When a client requests a certain bandwidth, the AP computes the collision probability by passively monitoring the channel, and computes the available bandwidth changing the CW/TXOP and check if the requested bandwidth fits. This method shares the same problem as [90] in that it guarantees bandwidth only. Also, the assumption of the analytical method that channels are always saturated is far from true in real environments, as we have seen in Chapter 5.

Sachin et al. [17] proposed a new metric for admission control, the channel utilization estimate (CUE), which is the fraction of time per time unit needed to transmit the flow over the network. The CUE is computed per flow using the average transmission rate measured for a short time and the average backoff measured at the AP, and total CUE is calculated by summing up the CUEs of all flows. Assuming that 15% of the total network capacity is wasted due to collisions, which is measured with 10 clients in their previous study, they use 0.85 as the maximum total CUE (CUETotalMax). Even if we assume the CUE is computed accurately, applying the fixed collision rate to CUETotalMax can degrade the QoS of VoIP or waste the bandwidth because according to our measurement results in a test-bed, the collision rate changes from 5% to 60% even with the same number of VoIP sources. Also, it is difficult to correctly estimate the QoS of a new flow using the remaining CUE value.

Zhai et al. [93] proposed a call admission scheme using the channel busyness ratio ( $R_b$ ), the ratio of the time that a channel is determined to be busy, which is similar to CUE. However, unlike CUE,  $R_b$  is computed in every client by looking at the actual MAC and PHY layer behavior. When a new call

Table 8.2: Comparison of CAC methods

Methods	Metric	Assumption	Adaptability (1)	Wasted BW	Extensibility (2)	802.11e
Theoretical [90][63][44]	CW/TXOP Computed BW	Saturated channel	Bad	<b>Low</b>	<b>Good</b>	<b>Applicable</b>
CUE/CBR [17][93]	CUE/CBR	Max CU	Bad (3)	Middle (3)	<b>Good</b>	N/A
Actual probing [49]	Delay packet loss	<b>No</b>	<b>Good</b>	High	Bad	N/A
QP-CAT	Queue size of the AP	<b>No</b>	<b>Good</b>	<b>Low</b>	<b>Good</b>	<b>Supported</b>

(1) Adaptability how they adapt to the change of channel status in real time

(2) Extensibility: how they can be extended to check multiple number or types of VoIP flows

(3) Due to the fixed MaxCU (= 0.85, 15% could be wasted)

is requested, the transmission rate is changed to the average channel utilization ( $CU$ ) and peak channel utilization ( $CU_{peak}$ ) and they are sent to the AP. Then, the AP computes the total  $CU$  and  $CU_{peak}$  and compare it with the maximum CU, which was measured in advance. However, the maximum CU varies according to the traffic type and channel condition and the wrong maximum CU wastes bandwidth or impairs the QoS. Also, according to their simulation results, 10% of the resources were wasted after the admission control, which shows the inefficiency of the call admission control algorithm.

Kuo et al. [44] used an analytical model to decide the admission of a new call. When a new call is requested, the expected bandwidth and delay are computed using an analytical model. However, the assumptions used in the analytical model have the same problem as those in [63].

Rosario et al. [19] proposed another analytical model based call admission control algorithm, which uses the channel occupancy time ( $T_{occ}$ ) computed using an existing IEEE 802.11e model. When  $T_{occ}$  is larger than the packetization interval of VoIP traffic, then further calls are rejected. Here, the 802.11e model that they used assumes non-saturated channel, and thus this method would be more realistic than other theoretical approaches. However, they tested the performance only via simulations using ideal environments without any background traffic even though their model is for 802.11e.

### 8.7.1 Comparison with other CAC methods

Table 8.2 compares the call admission methods with QP-CAT according to several criteria. They were evaluated from three points of view, namely, adaptability, channel utilization (waste of bandwidth), and extensibility.

**Adaptability:** Adaptability evaluates how the method adapts to the change of environments or channel status in real time. The theoretical approaches are not so adaptive to the changes because mostly their models are based on some ideal environments. CUE/CBR approaches also do not because the maximum CUE/CBR values are measured in advance in an environment. QP-CAT and probing methods adapt to the changes in real time because they measure the current channel status.

**Utilization of channel (Waste of BW):** Theoretical approaches do not waste any bandwidth, assuming that they work appropriately as in the ideal environments. CUE/CBR slightly waste bandwidth because they reserve some amount of bandwidth for collisions, which do not happen always. Actual probing method wastes bandwidth by that of the probing flows. QP-CAT has no such overhead.

**Extensibility:** Theoretical approaches, CUE/CBR, and QP-CAT can check the admissions of multiple number of new VoIP flows without any overhead, while in the actual probing method, the waste of the bandwidth increases as the number of probing flows increases.

## 8.8 Conclusion

I have proposed a novel call admission control in IEEE 802.11 WLANs, called QP-CAT, which predicts the impact of new VoIP calls accurately and allows the AP to make accurate admission decisions, protecting the QoS of existing VoIP flows and minimizing wasted bandwidth. It uses as the metric the queue size of the AP, which strongly correlates with the downlink delay as shown theoretically and experimentally. It predicts the queue size of the AP before new VoIP flows are admitted, by computing the number of additionally transmittable frames using CAT.

In order to evaluate the performance, QP-CAT was implemented in the QualNet simulator 3.9 and also in the MadWifi wireless card driver, and we have shown via simulations and experiments in a test-bed that it can accurately predict the impact of the additional VoIP flow on the existing flows and that it can be easily implemented in commercial wireless cards.

Furthermore, QP-CAT can be easily extended to QP-CATe, which supports the IEEE 802.11e standard. QP-CATe can accurately predicts the effect of new VoIP flows and background traffic in IEEE 802.11e. Also, multiple QP-CAT processes can be executed serially or in parallel to support admission of multiple types of VoIP traffic and simultaneous admission decision of multiple VoIP flows.

Even though only VoIP traffic was used in this study, QP-CAT can be used for admission control of any other traffic like video traffic, if we know the behavior (packet interval and size) of new flows. I also believe that QP-CAT can predict the change of the background traffic throughput and delay of real time traffic according to the access category of new flows in 802.11e, so that we can choose the best access category for new flows to maximize the total throughput while minimizing the delay increase.

# Chapter 9

## Conclusion

While the usage of VoIP traffic in 802.11 wireless networks is expected to increase in the near future, the quality of service for VoIP traffic is still an unsolved problem. In this thesis, sources of the QoS problem were investigated, and solutions were proposed, dividing the problems into three areas, namely, mobility, capacity, and call admission control.

### 9.1 QoS for user mobility

When users move around with mobile devices, handoffs occur between APs, and communication is disrupted during handoffs. When handoffs cause a change of subnet, the mobile devices need to acquire new IP addresses in the new subnet and update the existing sessions, which causes an extended network connectivity disruption. To reduce the network connectivity disruption time during handoffs, I proposed a new handoff procedure and architecture, namely, Selective Scanning and Caching for fast layer 2 handoff (Chapter 2), and seamless layer 3 handoff using the temporary IP address (Chapter 3) and pDAD (Chapter 4) in DHCP.

Table 9.1 shows the total handoff time using combined solutions. We can see that total layer 2 and 3 handoff time decreases from a few seconds to only 12 ms when Caching for layer 2 handoff and pDAD for layer 3 handoff are used.

Many papers have proposed new algorithms to reduce the handoff time between APs, and a few algorithms would work better than Selective Scanning and Caching in specific situations. However, I believe that Selective Scanning and Caching algorithm is still the best practical solution for seamless layer 2 handoff in that it requires only changes in client wireless card drivers, and it is very simple to implement while achieving significant improvement.

Table 9.1: Total handoff time using combined solutions

		Layer 2 handoff	
		Selective Scanning	Caching
No layer 3 handoff		130 ms	4 ms
Layer 3 handoff (with fast subnet detection)	Temporary IP address	234 ms	138 ms
	pDAD	108 ms	12 ms

Currently, IEEE 802.11 Working Group R (802.11r) is working to standardize fast roaming between APs, and the 802.11r standard will be released in 2008. It is mostly focused on the fast handoff with security enabled. Therefore, 802.11r can be combined with Selective Scanning and Caching for seamless layer 2 handoffs when security is enabled.

As mentioned in Chapter 1, Mobile IP was proposed in 1996 first to support seamless IP mobility, and new RFCs for Mobile IP (RFC 3344 [60] for IPv4 and RFC 3775 [34] for IPv6) were proposed in 2002 and 2004. However, regardless of the efforts to improve mobile IP, it has not been widely deployed yet. Also, the partial deployment is not useful because it works only when both clients and infrastructure and both home and visited networks support mobile IP. Unlike mobile IP, the solutions proposed in this thesis requires the changes of only either client or infrastructure of visited network, rather than both; the client side solution, layer 3 handoff using the temporary IP address, requires the change in clients, and the modified clients can achieve fast IP layer handoff in any visited network. Also, the server side solution, pDAD, requires change in visited networks, and any clients can acquire new IP addresses very quickly, reducing the IP layer handoff time. When both clients and server side can be modified, the combined solution with fast subnet change discovery and pDAD would work perfectly for seamless IP layer handoff.

## 9.2 VoIP capacity

Beyond the handoff problem, the QoS of VoIP traffic in WLANs degrades mostly due to the limited resources, and thus, increasing the capacity for VoIP traffic improves QoS. In this thesis, I proposed two improved MAC protocols, APC based on DCF and DPCF based on PCF. APC improves the capacity by balancing the uplink and downlink delay using CFT. DPCF eliminates the polling overhead caused in VBR VoIP traffic and improves capacity by up to 30%.

While most of the new MAC protocols to improve the VoIP capacity requires the changes in both clients and the APs, APC and DPCF require the changes only in the AP. Thus, any client associated with the AP using APC or DPCF can experience the better QoS. Therefore, the administrator can selectively upgrade the firmware or drivers of the APs, for example, only in crowded areas such as conference rooms and hotel lobbies.

Also, APC can be used to achieve fair resource distribution among clients. In this study, APC was applied only at the AP to control the downlink transmissions for the fairness between uplink and downlink. However, APC can be applied to all clients for the fairness among clients by controlling the uplink transmissions. Clients can overhear the uplink packets from other clients, extract the queue length of other clients, and control their transmission rate using the ratio between their own queue length and the average queue length of other clients. Also, using the PID control, all the nodes including the AP can control their transmission rate targeting 60 ms delay, and then all nodes including the AP achieve fair resource distribution.

DPCF also can be combined with the HCCA MAC layer in 802.11e. Even though this thesis showed that PCF works better than DCF for VoIP traffic and the problems of PCF for VBR VoIP traffic was eliminated in DPCF, PCF is not implemented in most of the commercial wireless cards. For the reason, the performance of the DPCF was not verified experimentally since PCF cannot be implemented in the wireless card driver. However, when HCCA is implemented in wireless cards in the future, we can apply the algorithm of DPCF to HCCA, and we can improve the performance of HCCA significantly by minimizing unnecessary polls and Null Function frames.

### 9.3 Call admission control

Even though we significantly improve the capacity for VoIP traffic in IEEE 802.11 wireless networks, the QoS of VoIP traffic would degrade if the number of VoIP calls exceeds the capacity. Therefore, to prevent the QoS degradation of existing calls due to new calls, we need call admission control. In this thesis, I have proposed a new call admission control, called QP-CAT (Chapter 8).

Efficient call admission control in wireless networks is difficult because the capacity for VoIP traffic changes dynamically according to the channel condition as well as VoIP traffic types. Thus, many approaches reserve some amount of bandwidth for the case of the change of channel status. It is a safe and easy solution but wastes some amount of bandwidth, and the overall capacity decreases. Therefore, the ultimate goal in call admission control is to protect the QoS for VoIP traffic while fully utilizing the channel bandwidth.

QP-CAT uses as the metric the queue size of the AP, which can be directly converted to the delay of VoIP traffic as proved via experiments, and thus it can estimate the QoS of VoIP traffic precisely. In order to predict the impact of new VoIP calls, QP-CAT emulates the packet transmission of new VoIP calls by monitoring the current actual transmissions, considering virtual deferrals and collisions, and it computes the queue size increase incurred by the virtual new VoIP flows.

Generally, multiple types of VoIP traffic can be used in a BSS, and call admission control algorithms should be able to support any type of VoIP call. Thus, multiple QP-CAT processes can be executed in parallel for multiple types of VoIP traffic, without overhead. Also, to handle the admission decision of multiple new VoIP calls in a short period, more than one QP-CAT processes can be executed serially to predict the impact of more than one VoIP calls.

Furthermore, QP-CAT can be extended to support background traffic in IEEE 802.11e standard, and the extension is called QP-CATe. QP-CATe can compute the impact of new VoIP calls under the existing background traffic, by emulating the frame transmission in EDCA.

QP-CAT does not define any framework for call admission control, and the behavior of clients application and the AP is not also defined. The framework of general admission control in IEEE 802.11 wireless networks is already standardized in the IEEE 802.11e standard [26], and the call admission control protocol at the application layer such as resource reservation protocol (RFC3312 [7] in SIP [68]), Resource ReSerVation Protocol (RSVP) [5], or Next Step In Signaling (NSIS) [20] is also already defined. Thus, QP-CAT should be combined with such a MAC layer framework and application protocols for call admission control.

# Bibliography

- [1] I Aad and C Castelluccia. Differentiation mechanism for IEEE 802.11. In *IEEE INFOCOM*, pages 209–218, Anchorage, Alaska, April 2001.
- [2] N. Akhtar, M. Georgiades, C. Politis, and R. Tafazolli. SIP-based End System Mobility Solution for All-IP Infrastructures. In *IST Mobile & Wireless Communications Summit 2003*, Aveiro, Portugal, June 2003.
- [3] M Barry, A T Campbell, and A Veres. Distributed control algorithms for service differentiation in wireless packet networks. In *IEEE INFOCOM*, pages 582–590, Anchorage, Alaska, April 2001.
- [4] John C. Bisket. Bit-rate selection in wireless networks. Master’s thesis, MIT, 2005.
- [5] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin. Resource ReSerVation Protocol - Version 1 Functional Specification. RFC 2205, Internet Engineering Task Force, Sept 1997.
- [6] P. Brady. A Model for Generating On-Off Speech Patterns in Two-Way Conversation. *Bell Syst. Tech. Journal*, 48(7):2245–2272, Sept. 1969.
- [7] G. Camarillo, W. Marshall, and J. Rosenberg. Integration of Resource Management and Session Initiation Protocol (SIP). RFC 3312, IETF, Oct 2002.
- [8] Casetti, C. Chiasserini, and C.-F. Improving fairness and throughput for voice traffic in 802.11e EDCA. In *Personal, Indoor and Mobile Radio Communications (PIMRC)*, volume 1, pages 525–530, 2004.
- [9] D. Chen, S. Garg, M. Kappes, and K. S. Trivedi. Supporting VoIP Traffic in IEEE 802.11 WLAN with Enhanced Medium Access Control (MAC) for Quality of Service. Technical report, Avaya Labs Research, 2002.
- [10] Dongyan Chen, Sachin Garg, Martin Kappes, and Kishor S. Trivedi. Supporting VBR VoIP Traffic in IEEE 802.11 WLAN in PCF Mode. Technical report, Avaya Labs, 2002.
- [11] D J Deng, R S Chang, and A Veres. A priority scheme for IEEE 802.11 DCF access method. *IEICE Trans. Commun.*, E82-B(1):96 – 102, Oct 1999.
- [12] R. Droms. Dynamic Host Configuration Protocol (DHCP). RFC 2131, Mar 1997.
- [13] Ashutosh Dutta, Sunil Madhanie, Wai Chen, Onur Altintas, and Henning Schulzrinne. Optimized fast-handoff schemes for application layer mobility management. *SIGMOBILE Mobile Computing and Communication Review*, 7(1):17–19, 2003.
- [14] Ethereal Network Protocol Analyzer. <http://www.ethereal.com>.
- [15] Andrea G. Forte, Sangho Shin, and Henning Schulzrinne. Passive Duplicate Address Detection for Dynamic Host Configuration Protocol (DHCP). Internet Draft (work in progress), Nov 2005.
- [16] Gene F. Franklin, J. David Powell, and Abbas Emami-Naeini. *Feedback control of dynamic systems*. Addison Wesley, 1987.
- [17] Sachin Garg and M. Kappes. Admission control for VoIP traffic in IEEE 802.11 networks. In *GLOBECOM*, pages 3514–3518, San Francisco, Dec 2003.
- [18] Sachin Garg and Marin Kappes. An Experimental Study of Throughput for UDP and VoIP Traffic in IEEE 802.11b Networks. In *IEEE Wireless Communications and Networking Conference (WCNC)*, volume 3, pages 1748–1753, New Orleans, LA, 2003.
- [19] Rosario G. Garropo, Stefano Giordano, Stefano Lucetti, and Luca Tavanti. A Model-based Admission Control for IEEE 802.11e Networks. In *International Conference on Communication (ICC)*, pages 398–402, Glasgow, Scotland, June 2007.
- [20] R. Hancock, G. Karagiannis, J. Loughney, and S. Van den Bosch. Next Steps in Signaling (NSIS): Framework. RFC 4080, Internet Engineering Task Force, June 2005.



- [21] T. Hiraguri, T. Ichikawa, M. Iizuka, and M. Morikura. Novel Multiple Access Protocol for Voice over IP in Wireless LAN. *IEICE Trans. Commun.*, E85-B(10):2145 – 2152, Oct 2002.
- [22] D. P. Hole and F. A. Tobagi. Capacity of an IEEE 802.11b Wireless LAN supporting VoIP. In *IEEE International Conference on Communications (ICC)*, pages 196–201, Paris, France, 2004.
- [23] IEEE. *IEEE Std. 802.11, Wireless LAN Medium Access Control (MAC) and Physical (PHY) specifications*, 1999.
- [24] IEEE. *IEEE Std. 802.11, Wireless LAN Medium Access Control (MAC) and Physical (PHY) specifications: Further Higher Data Rate Extension in the 2.4 GHz Band*, 2003.
- [25] IEEE. *IEEE Std. 802.11f, IEEE Trial-Use Recommended Practice for Multi-Vendor Access Point Interoperability via an Inter-Access Point Protocol Across Distribution Systems Supporting IEEE 802.11 Operation*, Feb 2003.
- [26] IEEE. *IEEE Std. 802.11e, Medium Access Control (MAC) Enhancements for Quality of Service (QoS)*, Feb 2005.
- [27] IEEE. *IEEE Draft 2.0 802.11n, Wireless LAN Medium Access Control (MAC) and Physical (PHY) specifications: Enhancements for Higher Throughput*, Nov 2007.
- [28] Internet System Consortium (ISC). dhcp-3.0.3. <http://www.isc.org/index.pl?sw/dhcp/>.
- [29] ITU-T Recommendation G.114. *One-way Transmission Time*, 2003.
- [30] ITU-T Recommendation P.59. *Artificial Conversational Speech*, 1993.
- [31] Anshul Jain. Handoff Delay for 802.11b Wireless LANs. Technical report, University of Kentucky, 2003.
- [32] Raj Jain, Dah-Ming Chiu, and W Hawe. A quantative measure of fairness and discrimination for resource allocation in shared computer systems. Technical Report TR-301, DEC, Sept 1984.
- [33] Jiwoong Jeong, Sunghyun Choi, and Chong kwon Kim. Achieving Weighted Fairness between Uplink and Downlink in IEEE 802.11 DCF-Based WLANs. In *International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QShine)*, Orlando, Florida, August 2005.
- [34] D. Johnson, C. Perkins, and J. Arrko. Mobility Support in IPv6. RFC 3775, Internet Engineering Task Force, June 2004.
- [35] Takehiro Kawata, Sangho Shin, Andrea G. Forte, and Henning Schulzrinne. Using dynamic PCF to improve the capacity for VoIP traffic in IEEE 802.11 networks. In *IEEE Wireless Communications and Networking Conference (WCNC)*, volume 3, pages 13–17, New Orleans, LA, Mar 2005.
- [36] M. Kershaw. Kismet Wireless Network Sniffer. <http://www.kismetwireless.net>.
- [37] Hye-Soo Kim, Sang-Hee Park, Chun-Su Park, Jae-Won Ki, and Sung-Jae Ko. Selective Channel Scanning for Fast Handoff in Wireless LAN using Neighbor Graph. In *International Technical Conference on Circuits/Systems, Computer and Communications*, Miyagi, Japan, July 2004.
- [38] Sung Won Kim, Byung-Seo Kim, and Yuguang Fang. Downlink and Uplink Resource Allocation in IEEE 802.11 Wireless LANs. *IEEE Trans. on Vehicular Technology*, 54(1):320–327, Jan. 2005.
- [39] Wooseong Kim, Myungchul Kim, Kyounghee Lee, Chansu Yu, and Ben Lee. Link layer assisted mobility support using SIP for real-time multimedia communications. In *International Workshop on Mobility Management and Wireless Access Protocols (MobiWac)*, pages 127–129, Philadelphia, PA, USA, 2004.
- [40] Young-Jae Kim and Young-Joo Suh. Adaptive polling MAC schemes for IEEE 802.11 wireless LANs supporting Voice-over-IP (VoIP) services. *Wireless Communications and Mobile Computing*, 4:903–916, 2004.
- [41] Toshikazu Kodama and Yasuhiro Katsube. Voice Performance in WLAN Networks - An Experimental Study. In *GLOBECOM*, pages 3504–3508, San Francisco, CA, 2003.
- [42] Andreas Kosep and Adam Wolisz. Voice transmissions in IEEE 802.11 based access network. In *International Symposium on a World of Wireless Mobile and Multimedia Networks (WoWMoM)*, pages 24–33, Rome, Italy, July 2001.
- [43] Haris Kremo, Ivan Seskar, and Predrag Spasojevic. An ORBIT Testbed Study of 802.11b DCF: Throughput, Latency, and the Capture Effect. In *IEEE Tridentcom 2006*, pages 308–309, Barcelona, Spain, Mar 2006.
- [44] Yu-Liang Kuo, Chi-Hung Lu, E.H.K. Wu, and Gen-Huey Chen. An admission control strategy for differentiated services in IEEE 802.11. In *GLOBECOM*, pages 707–712, San Francisco, Dec 2003.
- [45] Mathieu Lacage, Mohammad Hossein Manshaei, and Thierry Turletti. IEEE 802.11 rate adaptation: a practical approach. In *International Workshop on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*, pages 126–134, Venice, Italy, 2004. ACM Press.
- [46] Jouni Malinen. Host AP driver for Intersil Prism2/2.5/3. <http://hostap.epitest.fi>.

- [47] Anthony Mcauley, Subir Das, Shinichi Baba, and Yasuro Shobatake. Dynamic Registration and Configuration Protocol (DRCP). Internet draft (work in progress), Internet Engineering Task Force, July 2000.
- [48] Steven McCanne and Sally Floyd. ns Network Simulator. <http://www.isi.edu/nsnam/ns/>.
- [49] P. McGovern, S. Chung, S. Murphy, and L. Murphy. Endpoint Admission Control for VoIPoWLAN. In *International Conference on Telecommunication (ICT)*, Madeira Island, Portugal, May 2006.
- [50] David L. Mills. Network Time Protocol (Version 3) Specification, Implementation and Analysis. RFC 1305, Internet Engineering Task Force, March 1992.
- [51] Arunesh Mishra, Minh Shin, and William Arbaugh. An Empirical Analysis of the IEEE 802.11 MAC Layer Handoff Process. *ACM SIGCOMM Computer Communication Review*, 33(2):93–102, April 2003.
- [52] Arunesh Mishra, Minh Shin, and William Arbaugh. Context Caching using Neighbor Graphs for Fast Handoffs in a Wireless Network. Technical report, University of Maryland, February 2004.
- [53] Nick 'Sharkey' Moore. Optimistic Duplicate Address Detection for IPv6. Internet Draft, Dec 2005.
- [54] K. Nichols, K. Nichols, F. Baker, and D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. RFC 2474, IETF, Dec 1998.
- [55] ONOE rate control. <http://madwifi.org/wiki/UserDocs/RateControl>.
- [56] O.Tickoo and B.Sikdar. Queueing analysis and delay mitigation in IEEE 802.11 random access MAC based wireless networks. In *IEEE INFOCOM*, Hong Kong, 2004.
- [57] Sangheun Park and Yanghee Choi. Fast Inter-AP Handoff using Predictive-Authentication Scheme in a Public Wireless LAN. In *Networks 2002*, Atlanta, Georgia, August 2002.
- [58] Sangheun Park and Yanghee Choi. Pre-Authenticated Fast Handoff in a Public Wireless LAN based on IEEE 802.1x Mode. In *IFIP Personal Wireless Communications (PWC)*, Singapore, Oct 2002.
- [59] C. Perkins. IP Mobility Support. RFC 2002, Internet Engineering Task Force, October 1996.
- [60] C. Perkins. IP Mobility Support for IPv4. RFC 3344, Internet Engineering Task Force, August 2002.
- [61] S Pilosof, R Ramjee, D Raz, Y Shavitt, and P Sinha. Understanding TCP fairness over wireless LAN. In *IEEE INFOCOM*, pages 863–872, San Francisco, Mar 2003.
- [62] David C. Plummer. Ethernet Address Resolution Protocol. RFC 826, Internet Engineering Task Force, November 1982.
- [63] Dennis Pong and Tim Moors. Call Admission Control for IEEE 802.11 Contention Access Mechanism. In *GLOBECOM*, pages 174–178, San Francisco, Dec 2003.
- [64] QualNet Network Simulator 3.7. <http://www.scalable-networks.com>.
- [65] Ishwar Ramani and Stefan Savage. SyncScan: Practical Fast Handoff for 802.11 Infrastructure Networks. In *IEEE INFOCOM*, Miami, Florida, March 2005.
- [66] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh. Overview of the ORBIT Radio Grid Testbed for Evaluation of Next-Generation Wireless Network Protocols. In *IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1664–1669, New Orleans, LA, 2005.
- [67] Maua Rodrig, Charels Reis, Ratul Mahajan, David Wetherall, and John Zahorjan. Measurement-based characterization of 802.11 in a hotspot setting. In *ACM SIGCOMM Workshop on experimental approaches to wireless network design and analysis (E-WIND)*, pages 5–10, Philadelphia, PA, Aug 2005.
- [68] J. Rosenberg, Henning Schulzrinne, G. Camarillo, A. R. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. SIP: Session Initiation Protocol. RFC 3261, Internet Engineering Task Force, June 2002.
- [69] Scalable Network Technologies, Inc. *QualNet 3.7 User's Guide*, 2005.
- [70] Henning Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550, IETF, Jul 2003.
- [71] Sangho Shin, Andrea G. Forte, Anshuman Singh Rawat, and Henning Schulzrinne. Reducing MAC layer handoff latency in IEEE 802.11 wireless LANs. In *International Workshop on Mobility Management and Wireless Access Protocols (MobiWac)*, pages 19–26, Philadelphia, PA, USA, 2004.
- [72] Sangho Shin and Henning Schulzrinne. Balancing uplink and downlink delay of VoIP traffic in IEEE 802.11 WLANs using APC. In *International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QShine)*, Waterloo, Canada, Aug 2006.

- [73] Sangho Shin and Henning Schulzrinne. Experimental measurement of the capacity for VoIP traffic in IEEE 802.11 WLANs. In *IEEE INFOCOM*, Anchorage, Alaska, May 2007.
- [74] FirstHand Technology. <http://www.firsthandtech.com>.
- [75] N. Smavatkul, Y. Chen, and S. Emeott. Voice Capacity Evaluation of IEEE 802.11a with Automatic Rate Selection. In *GLOBECOM*, volume 1, pages 518–522, San Francisco, California, 2003.
- [76] T. Suzuki and S. Tasaka. Performance Evaluation of Priority-Based Multimedia Transmission with the PCF in an IEEE 802.11 Standard Wireless LAN. In *Personal, Indoor and Mobile Radio Communications (PIMRC)*, volume 2, pages G70–G77, San Diego, 2001.
- [77] Mineo Takai, Jay Martin, and Rajive Bagrodia. Effects of wireless physical layer modeling in mobile ad hoc networks. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 87–94, Long Beach, California, 2001.
- [78] Y. T'Joens, C. Hublet, and P. De Schrijver. DHCP reconfiguration extension. RFC 3202, IETF, December 2001.
- [79] TTCP Utility. <http://www.pcausa.com/Utilities/pcattcp.htm>.
- [80] Columbia University. Columbia SIP user agent (SIPc). <http://www.cs.columbia.edu/IRT/sipc>.
- [81] Nitin H Vaidya. Weak duplicate address detection in mobile ad hoc networks. In *ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc)*, pages 206–216, Lausanne, Switzerland, June 2002.
- [82] D. Vali, S. Paskalis, A. Kaloxylis, and L. Merakos. A SIP-based method for intra-domain handoffs. In *IEEE Vehicular Technology Conference (VTC)*, volume 3, pages 2068–2072, Orlando, Florida, Oct 2003.
- [83] M. Veeraraghavan, N. Cocker, and T. Moors. Support of voice services in IEEE 802.11 wireless LANs. In *IEEE INFOCOM*, volume 1, pages 488–497, Anchorage, Alaska, 2001.
- [84] Hector Velayos and Gunnar Karlsson. Techniques to Reduce IEEE 802.11b MAC Layer Handover Time. Technical report, Royal Institute of Technology, Stockholm, Sweden, April 2003.
- [85] Wei Wang, Soung Chang Liew, and V.O.K. Li. Solutions to performance problems in VoIP over a 802.11 wireless LAN. *IEEE Transactions on Vehicular Technology*, 54(1):366–384, Jan. 2005.
- [86] Xin Gang Wang, Geyong Min, Mellor, and J.E. Improving VoIP application's performance over WLAN using a new distributed fair MAC scheme. In *Advanced Information Networking and Applications (AINA)*, volume 1, pages 126–131, Hukuoka, Japan, March 2004.
- [87] Kilian Weniger. Passive Duplicate Address Detection in Mobile Ad hoc Networks. In *IEEE Wireless Communications and Networking Conference (WCNC)*, New Orleans, Louisiana, Mar 2003.
- [88] Sven Wietholter and Christian Hoene. Design and Verification of an IEEE 802.11e EDCF Simulation Model in ns-2.26. Technical Report TKN-03-019, Telecommunication Networks Group, Technische Universität Berlin, November 2003.
- [89] Haitao Wu, Kun Tan, Yongguang Zhang, and Qian Zhang. Proactive Scan: Fast Handoff with Smart Triggers for 802.11 Wireless LAN. In *IEEE INFOCOM*, Anchorage, Alaska, May 2007.
- [90] Yang Xiao and Haizhon Li. Evaluation of distributed admission control for the IEEE 802.11e EDCA. *IEEE Communications Magazine*, 42(9):S20–S24, Sept 2004.
- [91] Jing-Yuan Yeh and Chienhua Chen. Support of multimedia services with the IEEE 802.11 MAC protocol. In *IEEE International Conference on Communications (ICC)*, New York, New York, April 2002.
- [92] J. Yu, S. Choi, and J. Lee. Enhancement of VoIP over IEEE 802.11 WLAN via Dual Queue Strategy. In *IEEE International Conference on Communications (ICC)*, Paris, France, Jun 2004.
- [93] Hongqiang Zhai, Xiang Chen, and Yuguang Fang. A call admission and rate control scheme for multimedia support over IEEE 802.11 wireless LANs. In *International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QShine)*, pages 76–83, Dallas, Texas, Jan 2004.
- [94] E. Ziouva and T. Antonakopoulos. Efficient Voice Communications over IEEE 802.11 WLANs Using Improved PCF Procedures. In *International Network Conference (INC)*, Plymouth, UK, July 2002.